Lossy Compression

Guillaume Tochon

guillaume.tochon@lrde.epita.fr

LRDE, EPITA

The curse of noise for real signals

Let's take some (real) image and try to compress it with lossless algorithms:

8-bits grayscale image.

The curse of noise for real signals

Let's take some (real) image and try to compress it with lossless algorithms:

8-bits grayscale image.

What we would expect if we zoomed in an homogeneous area (the sky, for instance):

…	150	150	150	150	150	150	150	150	150	150	150	…

The curse of noise for real signals

Let's take some (real) image and try to compress it with lossless algorithms:

8-bits grayscale image.

What we would expect if we zoomed in an homogeneous area (the sky, for instance):

…												…
…	150	150	150	150	150	150	150	150	150	150	150	…

What we actually get:

…												…
…	148	150	149	151	150	150	153	151	148	147	150	…

→ Variations are imperceptible for the eye…
→ … but they annihilate lossless compression techniques!

Let's take some (real) image and try to compress it with lossless algorithms:

8-bits grayscale image.

$\Downarrow$

```
-rw-r--r-- 1 gtochon lrde 177K mars  28 18:00 grenoble.tiff.zip
-rw-r--r-- 1 gtochon lrde 177K mars  28 17:58 grenoble.tiff.gz
-rw-r--r-- 1 gtochon lrde 151K mars  28 17:58 grenoble.tiff.bz2
-rw-r--r-- 1 gtochon lrde 264K mars  28 17:58 grenoble.tiff
```

What we would expect if we zoomed in an homogeneous area (the sky, for instance):

⋯												⋯
⋯	150	150	150	150	150	150	150	150	150	150	150	⋯

What we actually get:

⋯	148	150	149	151	150	150	153	151	148	147	150	⋯

$\rightarrow$ Variations are imperceptible for the eye...
$\rightarrow$ ... but they annihilate lossless compression techniques!

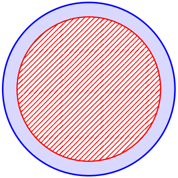
The curse of noise for real signals

Let's take some (real) image and try to compress it with lossless algorithms:

8-bits grayscale image.

⇓

```
-rw-r--r-- 1 gtochon lrde 177K mars  28 18:00 grenoble.tiff.zip
-rw-r--r-- 1 gtochon lrde 177K mars  28 17:58 grenoble.tiff.gz
-rw-r--r-- 1 gtochon lrde 151K mars  28 17:58 grenoble.tiff.bz2
-rw-r--r-- 1 gtochon lrde 264K mars  28 17:58 grenoble.tiff
-rw-r--r-- 1 gtochon lrde  64K mars  28 18:00 grenoble.jpg
```



Non-compressible part of the whole image.

What we would expect if we zoomed in an homogeneous area (the sky, for instance):

···											···
	150	150	150	150	150	150	150	150	150	150	150	

What we actually get:

···											···
	148	150	149	151	150	150	153	151	148	147	150	

→ Variations are imperceptible for the eye...

→ ... but they annihilate lossless compression techniques!

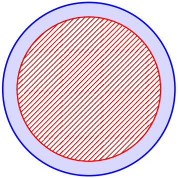
The curse of noise for real signals

Let's take some (real) image and try to compress it with lossless algorithms:

8-bits grayscale image.

$\Downarrow$

```
-rw-r--r-- 1 gtochon lrde 177K mars  28 18:00 grenoble.tiff.zip
-rw-r--r-- 1 gtochon lrde 177K mars  28 17:58 grenoble.tiff.gz
-rw-r--r-- 1 gtochon lrde 151K mars  28 17:58 grenoble.tiff.bz2
-rw-r--r-- 1 gtochon lrde 264K mars  28 17:58 grenoble.tiff
-rw-r--r-- 1 gtochon lrde  64K mars  28 18:00 grenoble.jpg
```



Non-compressible part of the whole image.

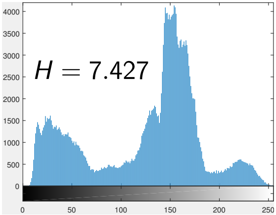
What we would expect if we zoomed in an homogeneous area (the sky, for instance):

…	150	150	150	150	150	150	150	150	150	150	150	…

What we actually get:

…	148	150	149	151	150	150	153	151	148	147	150	…

→ Variations are imperceptible for the eye…
→ … but they annihilate lossless compression techniques!



Lossy compression

Data before compression and after decompression are not the same.

Lossy compression algorithms:

- irremediably degrades the data by removing some part of the information.
  - → degradation is expected not to be noticeable by the end-user.
- are more efficient than lossless approaches in terms of compression ratio.
  - → at the expense of the commited inaccuracies during decompression.
- are most commonly used to compress multimedia data.
  - → noise is efficiently handled.

Lossy compression

Lossy compression

Data before compression and after decompression are not the same.

Lossy compression algorithms:

- irremediably degrades the data by removing some part of the information.
  - $\rightarrow$ degradation is expected not to be noticeable by the end-user.
- are more efficient than lossless approaches in terms of compression ratio.
  - $\rightarrow$ at the expense of the commited inaccuracies during decompression.
- are most commonly used to compress multimedia data.
  - $\rightarrow$ noise is efficiently handled.

Lossy compression $\equiv$ trade-off between performance and degradation.

Downsampling

A naive compression scheme

Let's take some grayscale image $\mathcal{I}$ with M rows and N columns (that is, some matrix $[\mathcal{I}(m,n)]_{M,N}$) and crudely downsample it by a factor of r ($\mathcal{I}_{\downarrow r} = \mathcal{I}(0{:}r{:}M\text{-}1, 0{:}r{:}N\text{-}1)$).

$\mathcal{I}$

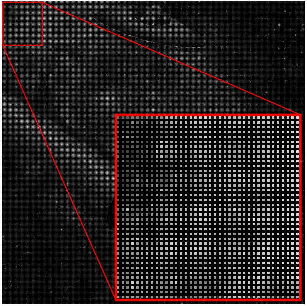
Let's take some grayscale image $\mathcal{I}$ with M rows and N columns (that is, some matrix $[\mathcal{I}(m,n)]_{M,N}$) and crudely downsample it by a factor of r ($\mathcal{I}_{\downarrow r} = \mathcal{I}(0{:}r{:}M\text{-}1, 0{:}r{:}N\text{-}1)$).

$\mathcal{I}$

$\mathcal{I}_{\downarrow 2}$

Let's take some grayscale image $\mathcal{I}$ with M rows and N columns (that is, some matrix $[\mathcal{I}(m,n)]_{M,N}$) and crudely downsample it by a factor of r ($\mathcal{I}_{\downarrow r} = \mathcal{I}(0{:}r{:}M\text{-}1, 0{:}r{:}N\text{-}1)$).

$\mathcal{I}$



$\mathcal{I}_{\downarrow 2}$

Let's take some grayscale image $\mathcal{I}$ with M rows and N columns (that is, some matrix $[\mathcal{I}(m,n)]_{M,N}$) and crudely downsample it by a factor of r ($\mathcal{I}_{\downarrow r} = \mathcal{I}(0{:}r{:}M\text{-}1, 0{:}r{:}N\text{-}1)$).

$\mathcal{I}$

$\mathcal{I}_{\downarrow 2}$

Downsampling

A naive compression scheme

Let's take some grayscale image $\mathcal{I}$ with M rows and N columns (that is, some matrix $[\mathcal{I}(m,n)]_{M,N}$) and crudely downsample it by a factor of r ($\mathcal{I}_{\downarrow r} = \mathcal{I}(0{:}r{:}M\text{-}1, 0{:}r{:}N\text{-}1)$).

$\mathcal{I}$

$\mathcal{I}_{\downarrow 2}$

$\mathcal{I}_{\downarrow 4}$

Downsampling

A naive compression scheme

Let's take some grayscale image $\mathcal{I}$ with M rows and N columns (that is, some matrix $[\mathcal{I}(m,n)]_{M,N}$) and crudely downsample it by a factor of r ($\mathcal{I}_{\downarrow r} = \mathcal{I}(0{:}r{:}M\text{-}1, 0{:}r{:}N\text{-}1)$).

$\mathcal{I}$

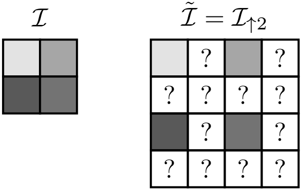
$\mathcal{I}_{\downarrow 2}$

$\mathcal{I}_{\downarrow 4}$

```
-rw-r--r-- 1 gtochon lrde 245K avril 20 13:25 randompic.tif
-rw-r--r-- 1 gtochon lrde  62K avril 20 13:25 randompic_down2.tif
-rw-r--r-- 1 gtochon lrde  16K avril 20 13:25 randompic_down4.tif
```

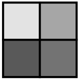
It's super effective!

If compression $\equiv$ downsampling, then decompression $\equiv$ upsampling $\equiv$ interpolation.

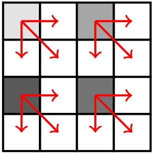


If compression $\equiv$ downsampling, then decompression $\equiv$ upsampling $\equiv$ interpolation.

$\mathcal{I}$



$\tilde{\mathcal{I}}_{NN}$



Nearest-neighbor interpolation:

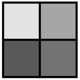
Copy in an unknown pixel the value of the *closest* known pixel (closest $\equiv$ leftmost and upmost closest pixel).

Upsampling

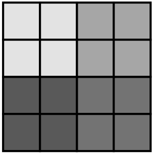
When things are getting tricky

If compression $\equiv$ downsampling, then decompression $\equiv$ upsampling $\equiv$ interpolation.

$\mathcal{I}$



$\tilde{\mathcal{I}}_{NN}$

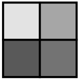


Nearest-neighbor interpolation:

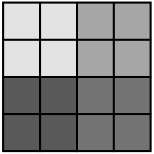
Copy in an unknown pixel the value of the *closest* known pixel (closest $\equiv$ leftmost and upmost closest pixel).

If compression $\equiv$ downsampling, then decompression $\equiv$ upsampling $\equiv$ interpolation.

$\mathcal{I}$



$\tilde{\mathcal{I}}_{NN}$



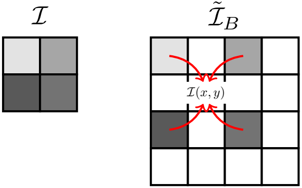
Nearest-neighbor interpolation:

Copy in an unknown pixel the value of the *closest* known pixel (closest $\equiv$ leftmost and upmost closest pixel).

👎 👎 blocking effect 👎 👎

$$\mathcal{I} \xrightarrow{\text{downsampling}} \mathcal{I}_{\downarrow 4} \xrightarrow[\text{nearest-neighbor}]{\text{interpolation}} \tilde{\mathcal{I}}_{NN}$$

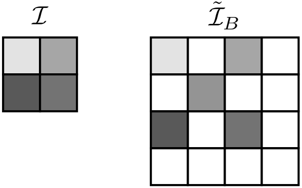
If compression $\equiv$ downsampling, then decompression $\equiv$ upsampling $\equiv$ interpolation.



Bilinear interpolation:

Write the pixel value as a bilinear function of its position
$\rightarrow \mathcal{I}(x,y) = \alpha x + \beta y + \gamma xy + \delta$.
Use the 4 closest known pixels to determine α, β, γ and δ and plug in previous equation.

If compression $\equiv$ downsampling, then decompression $\equiv$ upsampling $\equiv$ interpolation.



Bilinear interpolation:

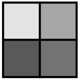
Write the pixel value as a bilinear function of its position $\rightarrow \mathcal{I}(x, y) = \alpha x + \beta y + \gamma xy + \delta$.
Use the 4 closest known pixels to determine α, β, γ and δ and plug in previous equation.

Upsampling

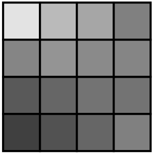
When things are getting tricky

If compression $\equiv$ downsampling, then decompression $\equiv$ upsampling $\equiv$ interpolation.

$\mathcal{I}$



$\tilde{\mathcal{I}}_B$



Bilinear interpolation:

Write the pixel value as a bilinear function of its position $\rightarrow \mathcal{I}(x, y) = \alpha x + \beta y + \gamma xy + \delta$.

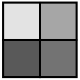
Use the 4 closest known pixels to determine α, β, γ and δ and plug in previous equation.

Upsampling

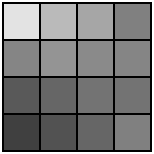
When things are getting tricky

If compression $\equiv$ downsampling, then decompression $\equiv$ upsampling $\equiv$ interpolation.

$\mathcal{I}$



$\tilde{\mathcal{I}}_B$



Bilinear interpolation:

Write the pixel value as a bilinear function of its position $\rightarrow \mathcal{I}(x, y) = \alpha x + \beta y + \gamma xy + \delta$.

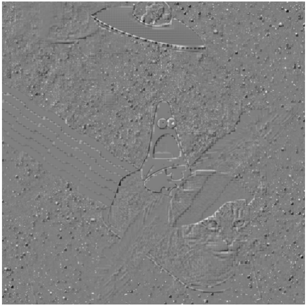
Use the 4 closest known pixels to determine α, β, γ and δ and plug in previous equation.

👎👎 blurring effect 👎👎

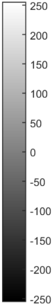
$$\mathcal{I} \xrightarrow{\text{downsampling}} \mathcal{I}_{\downarrow 4} \xrightarrow[\text{bilinear}]{\text{interpolation}} \tilde{\mathcal{I}}_B$$

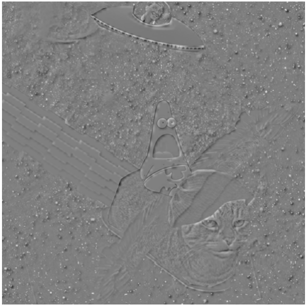
It sure looks bad...

... but how bad?



$\epsilon_{NN} = \mathcal{I} - \tilde{\mathcal{I}}_{NN}$





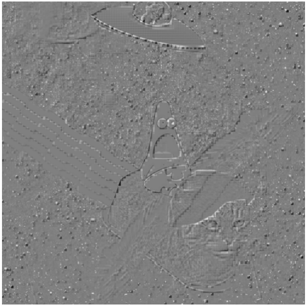
$\epsilon_{B} = \mathcal{I} - \tilde{\mathcal{I}}_{B}$

It sure looks bad...

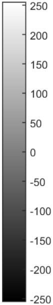
... but how bad?

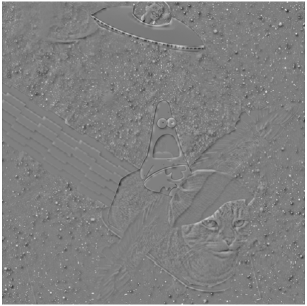
It is possible to quantitatively assess the compression/decompression error $\epsilon = \mathcal{I} - \tilde{\mathcal{I}}$ with numerical measures:

- Root mean square error: $\text{RMSE}(\epsilon) = \sqrt{\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\epsilon(m,n)^2}$
- Signal-to-noise ratio: $\text{SNR}(\epsilon) = 10\log\left(\frac{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\mathcal{I}(m,n)^2}{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\epsilon(m,n)^2}\right)$



$\epsilon_{NN} = \mathcal{I} - \tilde{\mathcal{I}}_{NN}$





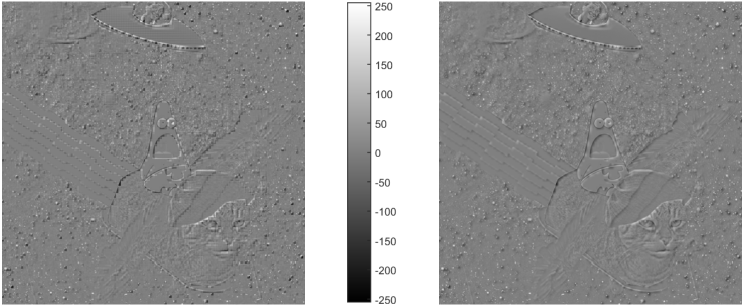
$\epsilon_B = \mathcal{I} - \tilde{\mathcal{I}}_B$

It sure looks bad...

... but how bad?

It is possible to quantitatively assess the compression/decompression error $\epsilon = \mathcal{I} - \tilde{\mathcal{I}}$ with numerical measures:

- Root mean square error: $\text{RMSE}(\epsilon) = \sqrt{\frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \epsilon(m,n)^2}$
- Signal-to-noise ratio: $\text{SNR}(\epsilon) = 10 \log \left(\frac{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \mathcal{I}(m,n)^2}{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \epsilon(m,n)^2} \right)$



$\epsilon_{NN} = \mathcal{I} - \tilde{\mathcal{I}}_{NN}$
$\text{RMSE}(\epsilon_{NN}) = 24.7$
$\text{SNR}(\epsilon_{NN}) = 12.9$ dB

$\epsilon_B = \mathcal{I} - \tilde{\mathcal{I}}_B$
$\text{RMSE}(\epsilon_B) = 20.7$
$\text{SNR}(\epsilon_B) = 14.5$ dB

Frequency decomposition of an image

Any image $\mathcal{I}$ can be expressed as the superposition of a *low-frequency* term $\mathcal{I}_{LF}$ and a *high-frequency* term $\mathcal{I}_{HF}$.

$\mathcal{I} = \mathcal{I}_{LF} + \mathcal{I}_{HF}$

$\mathcal{I}_{LF} \rightarrow$ Overall structure of the image: uniform areas, color gradients, slowly-varying components.

$\mathcal{I}_{HF} \rightarrow$ Details: sharp edges, rapidly-varying components, **noise**.

Any image $\mathcal{I}$ can be expressed as the superposition of a *low-frequency* term $\mathcal{I}_{LF}$ and a *high-frequency* term $\mathcal{I}_{HF}$.

$\mathcal{I} = \mathcal{I}_{LF} + \mathcal{I}_{HF}$

$\mathcal{I}_{LF} \rightarrow$ Overall structure of the image: uniform areas, color gradients, slowly-varying components.

$\mathcal{I}_{HF} \rightarrow$ Details: sharp edges, rapidly-varying components, **noise**.

Problem: How to discard the useless part of $\mathcal{I}_{HF}$ without removing any important information?

Frequency decomposition of an image

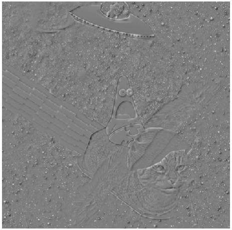
Any image $\mathcal{I}$ can be expressed as the superposition of a *low-frequency* term $\mathcal{I}_{LF}$ and a *high-frequency* term $\mathcal{I}_{HF}$.

$\mathcal{I}$

$=$

$\mathcal{I}_{LF}$

$+$



$\mathcal{I}_{HF}$

$\mathcal{I}_{LF} \rightarrow$ Overall structure of the image: uniform areas, color gradients, slowly-varying components.

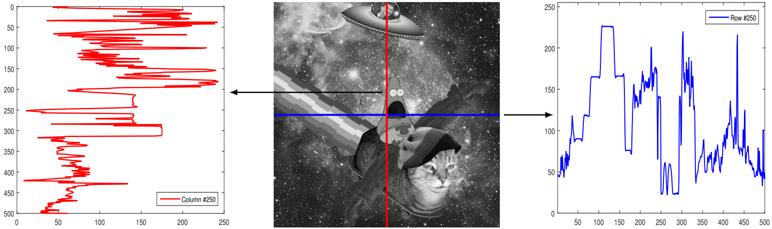
$\mathcal{I}_{HF} \rightarrow$ Details: sharp edges, rapidly-varying components, **noise**.

Problem: How to discard the useless part of $\mathcal{I}_{HF}$ without removing any important information? $\Rightarrow$ $\boxed{\text{Frequency analysis of the image}}$.

The *spatial* representation is not suited to carry out a frequency analysis of the image:

→ high frequencies are disseminated eveywhere in the image.

→ hard to tell whether a quick variation in terms of pixel intensity is due to an edge (important) or the noise (can be discarded).

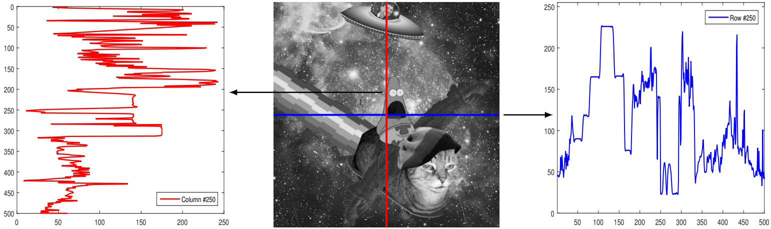


The *spatial* representation is not suited to carry out a frequency analysis of the image:

→ high frequencies are disseminated eveywhere in the image.

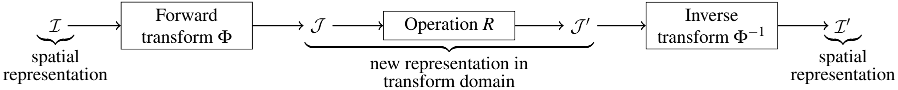
→ hard to tell whether a quick variation in terms of pixel intensity is due to an edge (important) or the noise (can be discarded).

We need a more suited representation to express the image in terms of frequencies

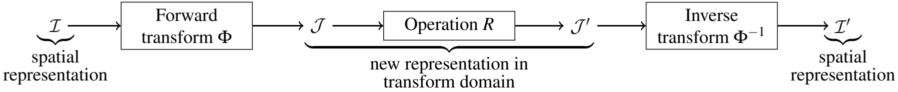


… and the need for a new one

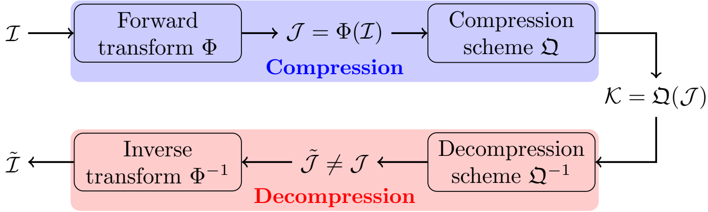
The image $\mathcal{I}$ is transformed into another image $\mathcal{J}$ through some reversible transform Φ such that all further processings to conduct are simpler/more efficient in the transform domain.



The image $\mathcal{I}$ is transformed into another image $\mathcal{J}$ through some reversible transform Φ such that all further processings to conduct are simpler/more efficient in the transform domain.



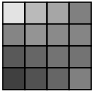
For lossy compression purposes:



The compression efficiency depends both on the used transform Φ/Φ^{-1} and the following compression/decompression scheme $\mathfrak{Q}/\mathfrak{Q}^{-1}$.

But wait...

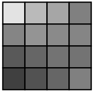
What did we exactly mean by spatial representation in the first place?



$$\equiv \begin{bmatrix} 227 & 186 & 166 & 127 \\ 133 & 148 & 138 & 133 \\ 89 & 102 & 115 & 115 \\ 64 & 82 & 148 & 127 \end{bmatrix} = \begin{bmatrix} \mathcal{I}(0,0) & \dots & \mathcal{I}(0,3) \\ \vdots & \ddots & \vdots \\ \mathcal{I}(3,0) & \dots & \mathcal{I}(3,3) \end{bmatrix} \equiv [\mathcal{I}(m,n)]_{4,4}$$

But wait...

What did we exactly mean by spatial representation in the first place?

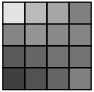


$$\equiv \begin{bmatrix} 227 & 186 & 166 & 127 \\ 133 & 148 & 138 & 133 \\ 89 & 102 & 115 & 115 \\ 64 & 82 & 148 & 127 \end{bmatrix} = \begin{bmatrix} \mathcal{I}(0,0) & \dots & \mathcal{I}(0,3) \\ \vdots & \ddots & \vdots \\ \mathcal{I}(3,0) & \dots & \mathcal{I}(3,3) \end{bmatrix} \equiv [\mathcal{I}(m,n)]_{4,4}$$

$$\equiv 227 \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 186 \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 166 \times \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \cdots + 127 \times \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

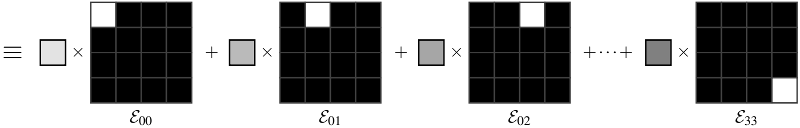
But wait...

What did we exactly mean by spatial representation in the first place?



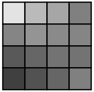
$$\equiv \begin{bmatrix} 227 & 186 & 166 & 127 \\ 133 & 148 & 138 & 133 \\ 89 & 102 & 115 & 115 \\ 64 & 82 & 148 & 127 \end{bmatrix} = \begin{bmatrix} \mathcal{I}(0,0) & \dots & \mathcal{I}(0,3) \\ \vdots & \ddots & \vdots \\ \mathcal{I}(3,0) & \dots & \mathcal{I}(3,3) \end{bmatrix} \equiv [\mathcal{I}(m,n)]_{4,4}$$

$$\equiv 227 \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 186 \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 166 \times \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \cdots + 127 \times \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



But wait...

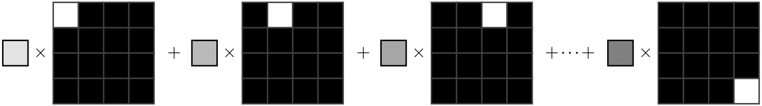
What did we exactly mean by spatial representation in the first place?



$$\equiv \begin{bmatrix} 227 & 186 & 166 & 127 \\ 133 & 148 & 138 & 133 \\ 89 & 102 & 115 & 115 \\ 64 & 82 & 148 & 127 \end{bmatrix} = \begin{bmatrix} \mathcal{I}(0,0) & \dots & \mathcal{I}(0,3) \\ \vdots & \ddots & \vdots \\ \mathcal{I}(3,0) & \dots & \mathcal{I}(3,3) \end{bmatrix} \equiv [\mathcal{I}(m,n)]_{4,4}$$

$$\equiv 227 \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 186 \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 166 \times \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \cdots + 127 \times \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\equiv$

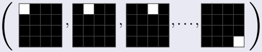


$\mathcal{E}_{00}$ $\mathcal{E}_{01}$ $\mathcal{E}_{02}$ $\mathcal{E}_{33}$

Spatial representation of an image $\mathcal{I}$

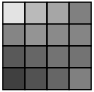
$\Leftrightarrow$

Decomposition of $\mathcal{I}$ over the set of basis images



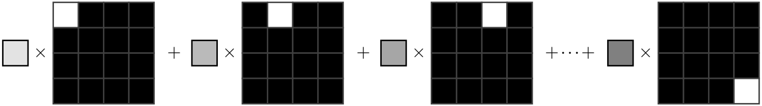
But wait...

What did we exactly mean by spatial representation in the first place?



$$\equiv \begin{bmatrix} 227 & 186 & 166 & 127 \\ 133 & 148 & 138 & 133 \\ 89 & 102 & 115 & 115 \\ 64 & 82 & 148 & 127 \end{bmatrix} = \begin{bmatrix} \mathcal{I}(0,0) & \dots & \mathcal{I}(0,3) \\ \vdots & \ddots & \vdots \\ \mathcal{I}(3,0) & \dots & \mathcal{I}(3,3) \end{bmatrix} \equiv [\mathcal{I}(m,n)]_{4,4}$$

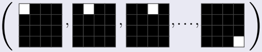
$$\equiv 227 \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 186 \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 166 \times \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \dots + 127 \times \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\equiv$ 

$\mathcal{E}_{00}$ $\mathcal{E}_{01}$ $\mathcal{E}_{02}$ $\mathcal{E}_{33}$

Spatial representation of an image $\mathcal{I}$

$\Leftrightarrow$

Decomposition of $\mathcal{I}$ over the set of basis images 

$\Leftrightarrow$

Pixel value $\mathcal{I}(m,n) \equiv$ coordinate with respect to the corresponding basis image $\mathcal{E}_{mn}$

Changing the representation domain

Changing the representation domain of $\mathcal{I}$

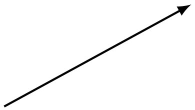
$\Leftrightarrow$

Changing the set of basis images and expressing $\mathcal{I}$ in this new basis

Changing the representation domain of $\mathcal{I}$

$\Leftrightarrow$

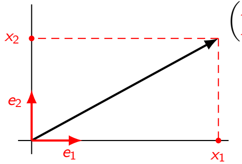
Changing the set of basis images and expressing $\mathcal{I}$ in this new basis



Changing the representation domain of $\mathcal{I}$

$\Leftrightarrow$

Changing the set of basis images and expressing $\mathcal{I}$ in this new basis

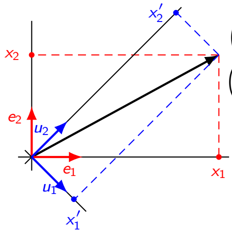


$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1 e_1 + x_2 e_2$$

Changing the representation domain of $\mathcal{I}$

$\Leftrightarrow$

Changing the set of basis images and expressing $\mathcal{I}$ in this new basis



$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1 e_1 + x_2 e_2$$

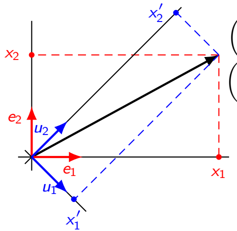
$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = x_1' u_1 + x_2' u_2$$

Changing the representation domain

Changing the representation domain of $\mathcal{I}$

$\Leftrightarrow$

Changing the set of basis images and expressing $\mathcal{I}$ in this new basis



$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1 e_1 + x_2 e_2$$

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = x_1' u_1 + x_2' u_2$$

$$= \begin{bmatrix} 200 & 150 \\ 50 & 100 \end{bmatrix} = \mathcal{I}$$

$= 200 \times$  $+ 150 \times$  $+ 50 \times$  $+ 100 \times$ 

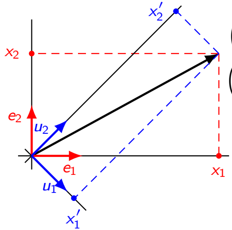
$\mathcal{E}_{00}$ $\mathcal{E}_{01}$ $\mathcal{E}_{10}$ $\mathcal{E}_{11}$

Changing the representation domain

Changing the representation domain of $\mathcal{I}$

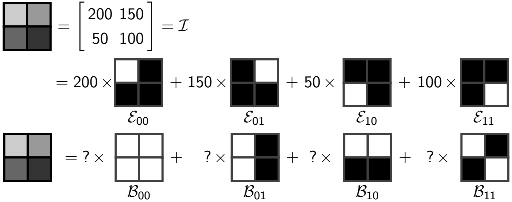
$\Leftrightarrow$

Changing the set of basis images and expressing $\mathcal{I}$ in this new basis



$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1 e_1 + x_2 e_2$$

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = x_1' u_1 + x_2' u_2$$



$$= \begin{bmatrix} 200 & 150 \\ 50 & 100 \end{bmatrix} = \mathcal{I}$$

$$= 200 \times \mathcal{E}_{00} + 150 \times \mathcal{E}_{01} + 50 \times \mathcal{E}_{10} + 100 \times \mathcal{E}_{11}$$

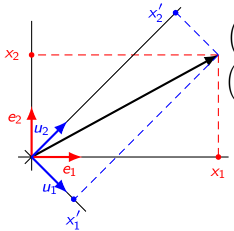
$$= ? \times \mathcal{B}_{00} + ? \times \mathcal{B}_{01} + ? \times \mathcal{B}_{10} + ? \times \mathcal{B}_{11}$$

Changing the representation domain

Changing the representation domain of $\mathcal{I}$

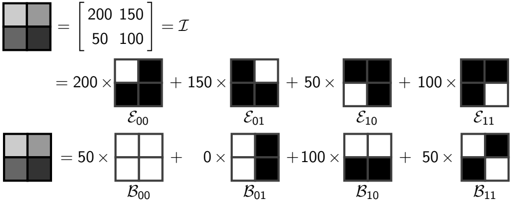
$\Leftrightarrow$

Changing the set of basis images and expressing $\mathcal{I}$ in this new basis



$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1 e_1 + x_2 e_2$$

$$\begin{pmatrix} x'_1 \\ x'_2 \end{pmatrix} = x'_1 u_1 + x'_2 u_2$$

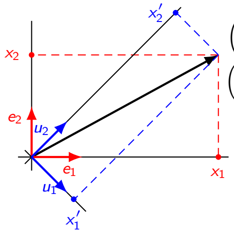


Changing the representation domain

Changing the representation domain of $\mathcal{I}$

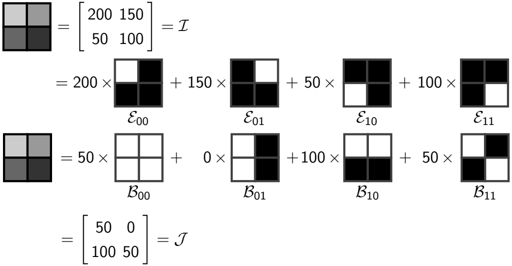
$\Leftrightarrow$

Changing the set of basis images and expressing $\mathcal{I}$ in this new basis



$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1 e_1 + x_2 e_2$$

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = x_1' u_1 + x_2' u_2$$



$$= \begin{bmatrix} 200 & 150 \\ 50 & 100 \end{bmatrix} = \mathcal{I}$$

$$= 200 \times \mathcal{E}_{00} + 150 \times \mathcal{E}_{01} + 50 \times \mathcal{E}_{10} + 100 \times \mathcal{E}_{11}$$

$$= 50 \times \mathcal{B}_{00} + 0 \times \mathcal{B}_{01} + 100 \times \mathcal{B}_{10} + 50 \times \mathcal{B}_{11}$$

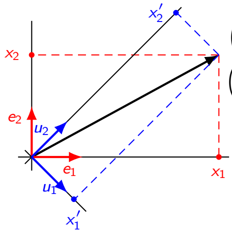
$$= \begin{bmatrix} 50 & 0 \\ 100 & 50 \end{bmatrix} = \mathcal{J}$$

Changing the representation domain

Changing the representation domain of $\mathcal{I}$

$\Leftrightarrow$

Changing the set of basis images and expressing $\mathcal{I}$ in this new basis



$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1 e_1 + x_2 e_2$$

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = x_1' u_1 + x_2' u_2$$

$$= \begin{bmatrix} 200 & 150 \\ 50 & 100 \end{bmatrix} = \mathcal{I}$$

$$= 200 \times \mathcal{E}_{00} + 150 \times \mathcal{E}_{01} + 50 \times \mathcal{E}_{10} + 100 \times \mathcal{E}_{11}$$

$$= 50 \times \mathcal{B}_{00} + 0 \times \mathcal{B}_{01} + 100 \times \mathcal{B}_{10} + 50 \times \mathcal{B}_{11}$$

$$= \begin{bmatrix} 50 & 0 \\ 100 & 50 \end{bmatrix} = \mathcal{J}$$

$$= \begin{bmatrix} 200 & 150 \\ 50 & 100 \end{bmatrix} = \mathcal{I} \text{ with respect to basis } \mathcal{E} = \left(\mathcal{E}_{00}, \mathcal{E}_{01}, \mathcal{E}_{10}, \mathcal{E}_{11} \right)$$

$$\equiv \begin{bmatrix} 50 & 0 \\ 100 & 50 \end{bmatrix} = \mathcal{J} \text{ with respect to basis } \mathcal{B} = \left(\mathcal{B}_{00}, \mathcal{B}_{01}, \mathcal{B}_{10}, \mathcal{B}_{11} \right)$$

Two-dimensional discrete linear transform (1/2)

Assume that $\mathcal{I}$ has M rows and N columns and call Φ the *linear* transformation that maps $\mathcal{I} = [\mathcal{I}(m,n)]_{M,N}$ into $\mathcal{J} = [\mathcal{J}(u,v)]_{M,N}$ $\rightarrow \mathcal{J} = \Phi(\mathcal{I})$.

$$\begin{tikzcd} \mathcal{I} \arrow[rr, bend left, "\Phi(\cdot)"] & & \mathcal{J} \arrow[ll, bend left, "\Phi^{-1}(\cdot)"] \end{tikzcd}$$

Two-dimensional discrete linear transform (1/2)

Assume that $\mathcal{I}$ has M rows and N columns and call Φ the *linear* transformation that maps $\mathcal{I} = [\mathcal{I}(m,n)]_{M,N}$ into $\mathcal{J} = [\mathcal{J}(u,v)]_{M,N}$ $\rightarrow \mathcal{J} = \Phi(\mathcal{I})$.

$$\begin{tikzcd} \mathcal{I} \arrow[r, bend left, "\Phi(\cdot)"] & \mathcal{J} \arrow[l, bend left, "\Phi^{-1}(\cdot)"] \end{tikzcd}$$

Most classical image transforms Φ are defined such that:

$$\boxed{\mathcal{J}(u,v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \mathcal{I}(m,n)\phi(m,n,u,v)}$$

where

- $\phi(m,n,u,v)$ is called the *forward transform kernel*,
- the variables $u = 0, \ldots, M-1$ and $v = 0, \ldots, N-1$ are called the *transform variables*.

Two-dimensional discrete linear transform (1/2)

Assume that $\mathcal{I}$ has M rows and N columns and call Φ the *linear* transformation that maps $\mathcal{I} = [\mathcal{I}(m,n)]_{M,N}$ into $\mathcal{J} = [\mathcal{J}(u,v)]_{M,N}$ $\rightarrow \mathcal{J} = \Phi(\mathcal{I})$.

$$\begin{tikzcd} \mathcal{I} \arrow[r, bend left, "\Phi(\cdot)"] & \mathcal{J} \arrow[l, bend left, "\Phi^{-1}(\cdot)"] \end{tikzcd}$$

Most classical image transforms Φ are defined such that:

$$\boxed{\mathcal{J}(u,v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \mathcal{I}(m,n)\phi(m,n,u,v)}$$

where

- $\phi(m,n,u,v)$ is called the *forward transform kernel*,
- the variables $u = 0, \ldots, M-1$ and $v = 0, \ldots, N-1$ are called the *transform variables*.

The inverse transform is given by

$$\boxed{\mathcal{I}(m,n) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \mathcal{J}(u,v)\psi(m,n,u,v)}$$

with $\psi(m,n,u,v)$ being the *inverse transform kernel*.

Two-dimensional discrete linear transform (2/2)

The transform kernels are said to be *separable* if

$$\phi(m,n,u,v) = \phi_M(m,u)\phi_N(n,v)$$
$$\psi(m,n,u,v) = \psi_M(m,u)\psi_N(n,v)$$

Two-dimensional discrete linear transform (2/2)

The transform kernels are said to be *separable* if

$$\phi(m,n,u,v) = \phi_M(m,u)\phi_N(n,v)$$
$$\psi(m,n,u,v) = \psi_M(m,u)\psi_N(n,v)$$

Ex: 2D discrete Fourier transform kernels:

$$\phi(m,n,u,v) = e^{-i2\pi(\frac{mu}{M}+\frac{nv}{N})} = e^{-i2\pi\frac{mu}{M}} e^{-i2\pi\frac{nv}{N}} = \phi_M(m,u)\phi_N(n,v)$$
$$\psi(m,n,u,v) = \frac{1}{MN}e^{i2\pi(\frac{mu}{M}+\frac{nv}{N})} = \frac{1}{M}e^{i2\pi\frac{mu}{M}} \frac{1}{N}e^{i2\pi\frac{nv}{N}} = \psi_M(m,u)\psi_N(n,v)$$

Two-dimensional discrete linear transform (2/2)

The transform kernels are said to be *separable* if

$$\phi(m,n,u,v) = \phi_M(m,u)\phi_N(n,v)$$
$$\psi(m,n,u,v) = \psi_M(m,u)\psi_N(n,v)$$

Ex: 2D discrete Fourier transform kernels:

$$\phi(m,n,u,v) = e^{-i2\pi(\frac{mu}{M}+\frac{nv}{N})} = e^{-i2\pi\frac{mu}{M}}e^{-i2\pi\frac{nv}{N}} = \phi_M(m,u)\phi_N(n,v)$$
$$\psi(m,n,u,v) = \frac{1}{MN}e^{i2\pi(\frac{mu}{M}+\frac{nv}{N})} = \frac{1}{M}e^{i2\pi\frac{mu}{M}}\frac{1}{N}e^{i2\pi\frac{nv}{N}} = \psi_M(m,u)\psi_N(n,v)$$

When the kernels are separable, the transform and inverse tranform can be compactly written in matrix form:

$$\boxed{\mathcal{J} = \phi_M \mathcal{I} \phi_N^T} \text{ and } \boxed{\mathcal{I} = \psi_M \mathcal{J} \psi_N^T = \phi_M^{-1} \mathcal{J} \phi_N^{-T}}$$

Two-dimensional discrete linear transform (2/2)

The transform kernels are said to be *separable* if

$$\phi(m,n,u,v) = \phi_M(m,u)\phi_N(n,v)$$
$$\psi(m,n,u,v) = \psi_M(m,u)\psi_N(n,v)$$

Ex: 2D discrete Fourier transform kernels:

$$\phi(m,n,u,v) = e^{-i2\pi(\frac{mu}{M}+\frac{nv}{N})} = e^{-i2\pi\frac{mu}{M}} e^{-i2\pi\frac{nv}{N}} = \phi_M(m,u)\phi_N(n,v)$$
$$\psi(m,n,u,v) = \frac{1}{MN} e^{i2\pi(\frac{mu}{M}+\frac{nv}{N})} = \frac{1}{M} e^{i2\pi\frac{mu}{M}} \frac{1}{N} e^{i2\pi\frac{nv}{N}} = \psi_M(m,u)\psi_N(n,v)$$

When the kernels are separable, the transform and inverse tranform can be compactly written in matrix form:

$$\boxed{\mathcal{J} = \boldsymbol{\phi}_M \mathcal{I} \boldsymbol{\phi}_N^T} \text{ and } \boxed{\mathcal{I} = \boldsymbol{\psi}_M \mathcal{J} \boldsymbol{\psi}_N^T = \boldsymbol{\phi}_M^{-1} \mathcal{J} \boldsymbol{\phi}_N^{-T}}$$

When $M = N$, the transform kernels are said to be *symmetric* is they are functionnally equivalent (*i.e.* $\phi(m,n,u,v) = \phi(m,u)\phi(n,v)$ and $\psi(m,n,u,v) = \psi(m,u)\psi(n,v)$), and

$$\boxed{\mathcal{J} = \boldsymbol{\phi} \mathcal{I} \boldsymbol{\phi}^T} \text{ and } \boxed{\mathcal{I} = \boldsymbol{\phi}^{-1} \mathcal{J} \boldsymbol{\phi}^{-T}}$$

Obtaining the new basis images from the old ones

Bases $\mathcal{E}$ and $\mathcal{B}$ are obviously linked through the mapping Φ, but how?

Bases $\mathcal{E}$ and $\mathcal{B}$ are obviously linked through the mapping Φ, but how?

Obtaining the new basis images from the old ones

Bases $\mathcal{E}$ and $\mathcal{B}$ are obviously linked through the mapping Φ, but how?

$$= \begin{bmatrix} 200 & 150 \\ 50 & 100 \end{bmatrix} = \mathcal{I} \text{ in basis } \mathcal{E}$$

$$\mathcal{I} = \sum_{m=0}^{N-1} \sum_{n=0}^{M-1} \mathcal{I}(m,n)\mathcal{E}_{mn}$$

$= 200 \times$  $+ 150 \times$  $+ 50 \times$  $+ 100 \times$ 

$\mathcal{E}_{00}$ $\mathcal{E}_{01}$ $\mathcal{E}_{10}$ $\mathcal{E}_{11}$

Obtaining the new basis images from the old ones

Bases $\mathcal{E}$ and $\mathcal{B}$ are obviously linked through the mapping Φ, but how?

 $= \begin{bmatrix} 200 & 150 \\ 50 & 100 \end{bmatrix} = \mathcal{I}$ in basis $\mathcal{E}$ $\equiv$ $\begin{bmatrix} 50 & 0 \\ 100 & 50 \end{bmatrix} = \mathcal{J}$ in basis $\mathcal{B}$

$= 200 \times$  $+ 150 \times$  $+ 50 \times$  $+ 100 \times$ 

$\mathcal{E}_{00}$ $\mathcal{E}_{01}$ $\mathcal{E}_{10}$ $\mathcal{E}_{11}$

$= 50 \times$  $+ 0 \times$  $+ 100 \times$  $+ 50 \times$ 

$\mathcal{B}_{00}$ $\mathcal{B}_{01}$ $\mathcal{B}_{10}$ $\mathcal{B}_{11}$

$$\mathcal{I} = \sum_{m=0}^{N-1} \sum_{n=0}^{M-1} \mathcal{I}(m,n)\mathcal{E}_{mn}$$
$$= \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} \mathcal{J}(u,v)\mathcal{B}_{uv}$$

Bases $\mathcal{E}$ and $\mathcal{B}$ are obviously linked through the mapping Φ, but how?

 $= \begin{bmatrix} 200 & 150 \\ 50 & 100 \end{bmatrix} = \mathcal{I}$ in basis $\mathcal{E} \equiv \begin{bmatrix} 50 & 0 \\ 100 & 50 \end{bmatrix} = \mathcal{J}$ in basis $\mathcal{B}$

$$\mathcal{I} = \sum_{m=0}^{N-1} \sum_{n=0}^{M-1} \mathcal{I}(m,n)\mathcal{E}_{mn}$$

$= 200 \times$  $+ 150 \times$  $+ 50 \times$  $+ 100 \times$ 

$\mathcal{E}_{00}$ $\mathcal{E}_{01}$ $\mathcal{E}_{10}$ $\mathcal{E}_{11}$

$$= \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} \mathcal{J}(u,v)\mathcal{B}_{uv}$$

$= 50 \times$  $+ 0 \times$  $+ 100 \times$  $+ 50 \times$ 

$\mathcal{B}_{00}$ $\mathcal{B}_{01}$ $\mathcal{B}_{10}$ $\mathcal{B}_{11}$

$$= \Phi^{-1}(\mathcal{J})$$

$= \Phi^{-1}\left(\begin{bmatrix} 50 & 0 \\ 100 & 50 \end{bmatrix}\right)$ in basis $\mathcal{E}$

Obtaining the new basis images from the old ones

Bases $\mathcal{E}$ and $\mathcal{B}$ are obviously linked through the mapping Φ, but how?

 $= \begin{bmatrix} 200 & 150 \\ 50 & 100 \end{bmatrix} = \mathcal{I}$ in basis $\mathcal{E}$ $\equiv$ $\begin{bmatrix} 50 & 0 \\ 100 & 50 \end{bmatrix} = \mathcal{J}$ in basis $\mathcal{B}$

$$\mathcal{I} = \sum_{m=0}^{N-1} \sum_{n=0}^{M-1} \mathcal{I}(m,n)\mathcal{E}_{mn}$$

$= 200 \times$  $+ 150 \times$  $+ 50 \times$  $+ 100 \times$ 

$\mathcal{E}_{00}$ $\mathcal{E}_{01}$ $\mathcal{E}_{10}$ $\mathcal{E}_{11}$

$$= \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} \mathcal{J}(u,v)\mathcal{B}_{uv}$$

$= 50 \times$  $+ \ 0 \times$  $+ 100 \times$  $+ \ 50 \times$ 

$\mathcal{B}_{00}$ $\mathcal{B}_{01}$ $\mathcal{B}_{10}$ $\mathcal{B}_{11}$

$$= \Phi^{-1}(\mathcal{J})$$

$= \Phi^{-1}\left(\begin{bmatrix} 50 & 0 \\ 100 & 50 \end{bmatrix}\right)$ in basis $\mathcal{E}$

$$= \Phi^{-1}\left(\sum_{u=0}^{N-1} \sum_{v=0}^{M-1} \mathcal{J}(u,v)\mathcal{E}_{uv}\right)$$

$= \Phi^{-1}\Big(50 \times$  $+ 0 \times$  $+ 100 \times$  $+ 50 \times$  $\Big)$

$\mathcal{E}_{00}$ $\mathcal{E}_{01}$ $\mathcal{E}_{10}$ $\mathcal{E}_{11}$

Obtaining the new basis images from the old ones

Bases $\mathcal{E}$ and $\mathcal{B}$ are obviously linked through the mapping Φ, but how?

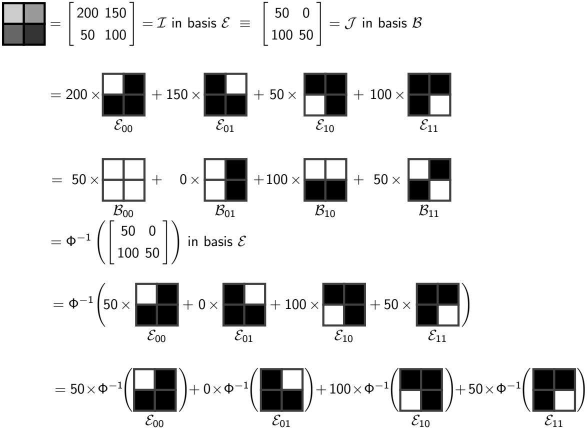
$$\mathcal{I} = \sum_{m=0}^{N-1}\sum_{n=0}^{M-1} \mathcal{I}(m,n)\mathcal{E}_{mn}$$

$$= \sum_{u=0}^{N-1}\sum_{v=0}^{M-1} \mathcal{J}(u,v)\mathcal{B}_{uv}$$

$$= \Phi^{-1}(\mathcal{J})$$

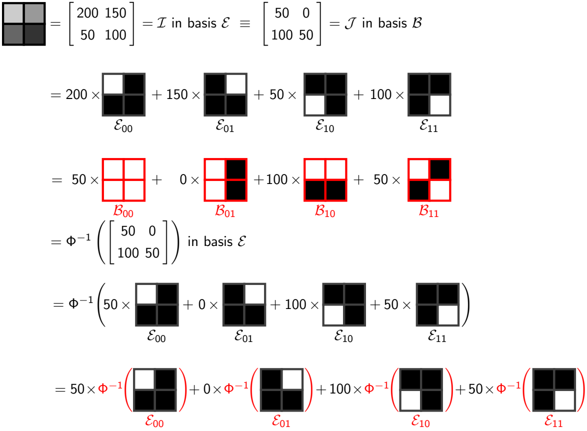
$$= \Phi^{-1}\left(\sum_{u=0}^{N-1}\sum_{v=0}^{M-1} \mathcal{J}(u,v)\mathcal{E}_{uv}\right)$$

$$= \sum_{u=0}^{N-1}\sum_{v=0}^{M-1} \mathcal{J}(u,v)\Phi^{-1}(\mathcal{E}_{uv})$$



Bases $\mathcal{E}$ and $\mathcal{B}$ are obviously linked through the mapping Φ, but how?

$$
\begin{aligned}
\mathcal{I} &= \sum_{m=0}^{N-1}\sum_{n=0}^{M-1} \mathcal{I}(m,n)\mathcal{E}_{mn} \\
&= \sum_{u=0}^{N-1}\sum_{v=0}^{M-1} \mathcal{J}(u,v)\color{red}{\mathcal{B}_{uv}} \\
&= \Phi^{-1}(\mathcal{J}) \\
&= \Phi^{-1}\left(\sum_{u=0}^{N-1}\sum_{v=0}^{M-1} \mathcal{J}(u,v)\mathcal{E}_{uv}\right) \\
&= \sum_{u=0}^{N-1}\sum_{v=0}^{M-1} \mathcal{J}(u,v)\color{red}{\Phi^{-1}(\mathcal{E}_{uv})}
\end{aligned}
$$

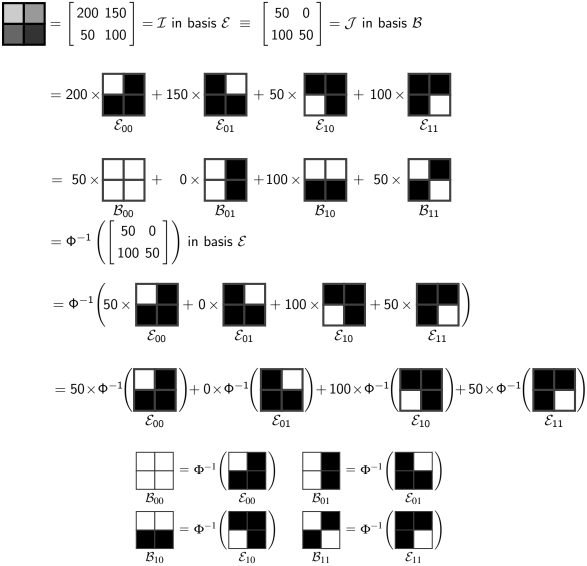


Obtaining the new basis images from the old ones

Bases $\mathcal{E}$ and $\mathcal{B}$ are obviously linked through the mapping Φ, but how?

$$
\begin{aligned}
\mathcal{I} &= \sum_{m=0}^{N-1}\sum_{n=0}^{M-1} \mathcal{I}(m,n)\mathcal{E}_{mn} \\
&= \sum_{u=0}^{N-1}\sum_{v=0}^{M-1} \mathcal{J}(u,v)\mathcal{B}_{uv} \\
&= \Phi^{-1}(\mathcal{J}) \\
&= \Phi^{-1}\left(\sum_{u=0}^{N-1}\sum_{v=0}^{M-1} \mathcal{J}(u,v)\mathcal{E}_{uv}\right) \\
&= \sum_{u=0}^{N-1}\sum_{v=0}^{M-1} \mathcal{J}(u,v)\Phi^{-1}(\mathcal{E}_{uv})
\end{aligned}
$$

$$\Rightarrow \boxed{\mathcal{B}_{uv} = \Phi^{-1}(\mathcal{E}_{uv})}$$



The Walsh-Hadamard transform

- → Named after Joseph Walsh and Jacques Hadamard.
- → Only works for square images of size $2^p \times 2^p$.
- → Can be implemented very efficiently (only requires addition and subtraction).

Joseph Walsh

Jacques Hadamard

The Walsh-Hadamard transform

- → Named after Joseph Walsh and Jacques Hadamard.
- → Only works for square images of size $2^p \times 2^p$.
- → Can be implemented very efficiently (only requires addition and subtraction).


Joseph Walsh


Jacques Hadamard

Walsh-Hadamard transform

Assuming that $N = 2^p$, a $N \times N$ image $\mathcal{I}$ and its Walsh-Hadamard transform $\mathcal{J}_H$ are linked by

$$\begin{aligned}\mathcal{J}_H(u,v) &= \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}(-1)^{\sum_{k=0}^{p-1} m_k u_k + n_k v_k}\,\mathcal{I}(m,n)\\ \mathcal{I}(m,n) &= \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}(-1)^{\sum_{k=0}^{p-1} u_k m_k + v_k n_k}\,\mathcal{J}_H(u,v)\end{aligned}$$

where

$$\begin{aligned} m &= \langle m_{p-1},\dots,m_1,m_0\rangle_2 = 2^{p-1}m_{p-1}+\cdots+2m_1+m_0\\ n &= \langle n_{p-1},\dots,n_1,n_0\rangle_2 = 2^{p-1}n_{p-1}+\cdots+2n_1+n_0\\ u &= \langle u_{p-1},\dots,u_1,u_0\rangle_2 = 2^{p-1}u_{p-1}+\cdots+2u_1+u_0\\ v &= \langle v_{p-1},\dots,v_1,v_0\rangle_2 = 2^{p-1}v_{p-1}+\cdots+2v_1+v_0\end{aligned}$$

are the base-2 representation of the indices m, n, u, v.

The Walsh-Hadamard kernel matrix

The Walsh-Hadamard transform $\mathcal{J}_{\mathbf{H}}$ (also called *Walsh-Hadamard spectrum*) of a $N \times N$ image $\mathcal{I}$ can be more conveniently defined thanks to the WHT kernel matrix $\mathbf{H}_N$ whose $(j,k)^{\text{th}}$ entry is

$$\mathbf{H}_N(j,k) = \frac{1}{\sqrt{N}}(-1)^{\sum_{n=0}^{p-1} j_n k_n}$$

The Walsh-Hadamard kernel matrix

The Walsh-Hadamard transform $\mathcal{J}_{\mathbf{H}}$ (also called *Walsh-Hadamard spectrum*) of a $N \times N$ image $\mathcal{I}$ can be more conveniently defined thanks to the WHT kernel matrix $\mathbf{H}_N$ whose $(j,k)^{\text{th}}$ entry is

$$\mathbf{H}_N(j,k) = \frac{1}{\sqrt{N}}(-1)^{\sum_{n=0}^{p-1} j_n k_n}$$

$$\mathbf{H}_1 = 1 \qquad \mathbf{H}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \mathbf{H}_4 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

The Walsh-Hadamard kernel matrix

The Walsh-Hadamard transform $\mathcal{J}_{\mathbf{H}}$ (also called *Walsh-Hadamard spectrum*) of a $N \times N$ image $\mathcal{I}$ can be more conveniently defined thanks to the WHT kernel matrix $\mathbf{H}_N$ whose $(j,k)^{\text{th}}$ entry is

$$\mathbf{H}_N(j,k) = \frac{1}{\sqrt{N}}(-1)^{\sum_{n=0}^{p-1} j_n k_n}$$

$$\mathbf{H}_1 = 1 \qquad \mathbf{H}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \mathbf{H}_4 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

And more generally

$$\forall N = 2^p,\ \mathbf{H}_{2N} = \frac{1}{\sqrt{2}}\begin{bmatrix} \mathbf{H}_N & \mathbf{H}_N \\ \mathbf{H}_N & -\mathbf{H}_N \end{bmatrix}$$

The Walsh-Hadamard kernel matrix

The Walsh-Hadamard transform $\mathcal{J}_{\mathbf{H}}$ (also called *Walsh-Hadamard spectrum*) of a $N \times N$ image $\mathcal{I}$ can be more conveniently defined thanks to the WHT kernel matrix $\mathbf{H}_N$ whose $(j,k)^{\text{th}}$ entry is

$$\mathbf{H}_N(j,k) = \frac{1}{\sqrt{N}}(-1)^{\sum_{n=0}^{p-1} j_n k_n}$$

$$\mathbf{H}_1 = 1 \qquad \mathbf{H}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \mathbf{H}_4 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

And more generally

$$\forall N = 2^p,\ \mathbf{H}_{2N} = \frac{1}{\sqrt{2}}\begin{bmatrix} \mathbf{H}_N & \mathbf{H}_N \\ \mathbf{H}_N & -\mathbf{H}_N \end{bmatrix}$$

Hadamard matrices are symmetric ($\mathbf{H}_N = \mathbf{H}_N^T$) and orthogonal ($\mathbf{H}_N^{-1} = \mathbf{H}_N^T$).

$\Rightarrow$ All put together $\begin{cases} \mathcal{J}_{\mathbf{H}} &= \mathbf{H}_N \mathcal{I} \mathbf{H}_N \\ \mathcal{I} &= \mathbf{H}_N \mathcal{J}_{\mathbf{H}} \mathbf{H}_N \end{cases}$

The Walsh-Hadamard transform for $N = 8$

The Walsh-Hadamard transform can be considered as a kind of Fourier transform
$\rightarrow$ suited to perform the frequency analysis of an image.

$$\mathbf{H}_8 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

The Walsh-Hadamard transform for $N = 8$

The Walsh-Hadamard transform can be considered as a kind of Fourier transform
$\rightarrow$ suited to perform the frequency analysis of an image.

For that, we must rearrange the rows of $\mathbf{H}_N$ in increasing number of sign changes.

$$\mathbf{H}_8 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

The Walsh-Hadamard transform for $N = 8$

The Walsh-Hadamard transform can be considered as a kind of Fourier transform
$\rightarrow$ suited to perform the frequency analysis of an image.

For that, we must rearrange the rows of $\mathbf{H}_N$ in increasing number of sign changes.

$$\mathbf{H}_8 = \frac{1}{2\sqrt{2}} \begin{array}{cc} & \begin{array}{c}\#\text{ sign}\\ \text{changes}\end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} & \begin{array}{c} \color{red}{0} \\ \color{red}{7} \\ \color{red}{3} \\ \color{red}{4} \\ \color{red}{1} \\ \color{red}{6} \\ \color{red}{2} \\ \color{red}{5} \end{array} \end{array}$$

The Walsh-Hadamard transform can be considered as a kind of Fourier transform
$\rightarrow$ suited to perform the frequency analysis of an image.

For that, we must rearrange the rows of $\mathbf{H}_N$ in increasing number of sign changes.

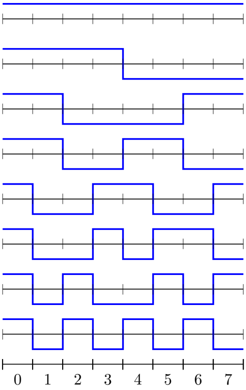
$$\mathbf{H}_8 = \frac{1}{2\sqrt{2}} \begin{array}{cc} & \begin{array}{c}\#\text{ sign}\\ \text{changes}\end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} & \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \end{array}$$

The Walsh-Hadamard transform for $N = 8$

The Walsh-Hadamard transform can be considered as a kind of Fourier transform $\rightarrow$ suited to perform the frequency analysis of an image.

For that, we must rearrange the rows of $\mathbf{H}_N$ in increasing number of sign changes.

$$\mathbf{H}_8 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

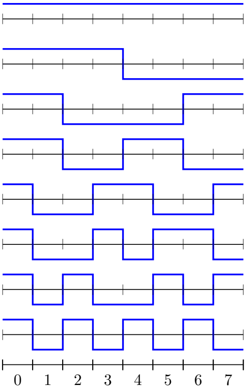


Walsh functions

The Walsh-Hadamard transform can be considered as a kind of Fourier transform
$\rightarrow$ suited to perform the frequency analysis of an image.

For that, we must rearrange the rows of $\mathbf{H}_N$ in increasing number of sign changes.

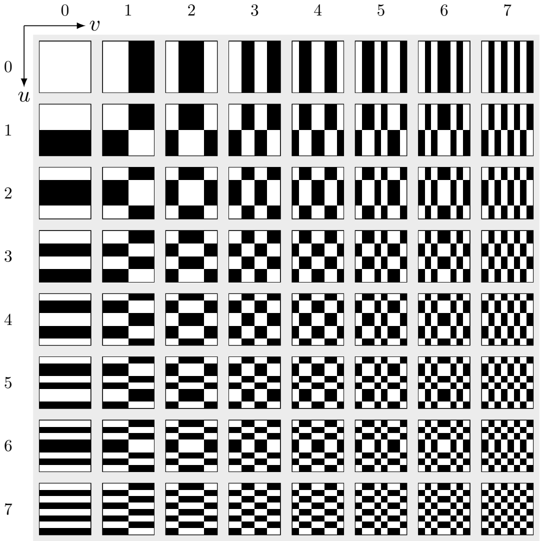
$$\mathbf{H}_8 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$



Walsh functions

Walsh-Hadamard transform $\equiv$ decomposition over the set of Walsh functions.

Walsh-Hadamard basis images for $N = 8$



Walsh-Hadamard basis images $\mathcal{B}_{uv}$ for $N = 8$. $\square = 1$, and $\blacksquare = -1$. The origin of each basis image is at its top-left corner.

Example

... finally!

Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.

Example

... finally!

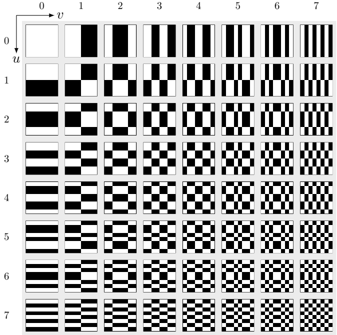
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.

$$\Rightarrow \mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Example

... finally!

Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

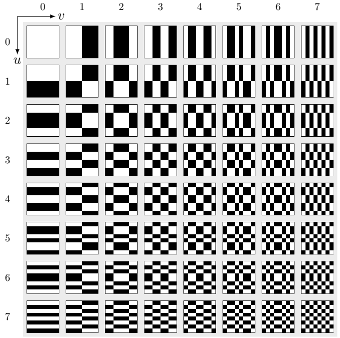
Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v) \mathcal{B}_{uv}$.

$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.

Example

... finally!

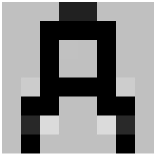
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8}\begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

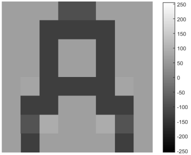
Theory gives $\mathcal{I} = \sum_{u=0}^{7}\sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u,v)\mathcal{B}_{uv}$.

$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$

$\tilde{\mathcal{I}}_1$

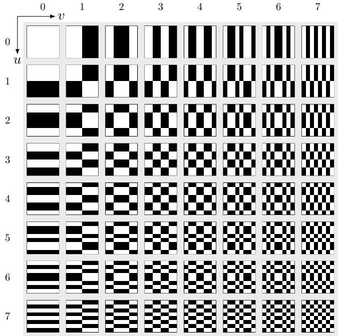


$\mathcal{I} - \tilde{\mathcal{I}}_1$

Example

... finally!

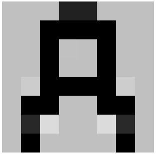
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

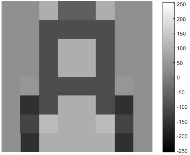
Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u,v) \mathcal{B}_{uv}$.

$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$

$\tilde{\mathcal{I}}_2$

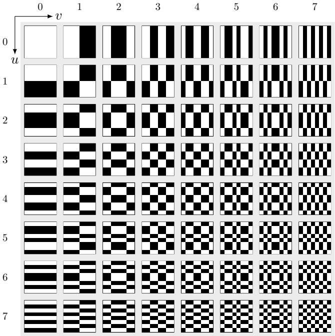


$\mathcal{I} - \tilde{\mathcal{I}}_2$

Example

... finally!

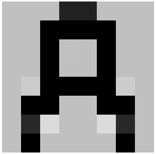
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



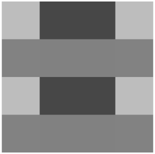
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8}\begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7}\sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u,v)\mathcal{B}_{uv}$.

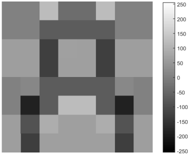
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$

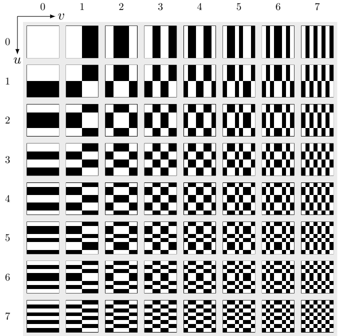


$\tilde{\mathcal{I}}_3$



$\mathcal{I} - \tilde{\mathcal{I}}_3$

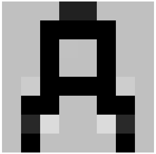
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



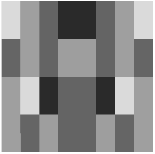
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v) \mathcal{B}_{uv}$.

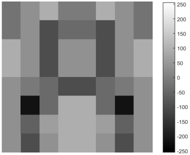
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$

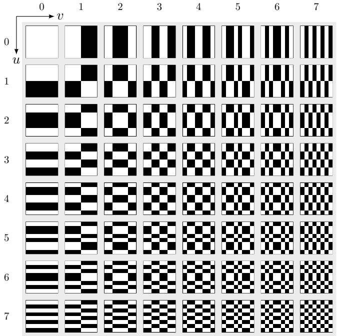


$\tilde{\mathcal{I}}_4$



$\mathcal{I} - \tilde{\mathcal{I}}_4$

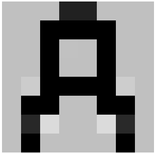
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



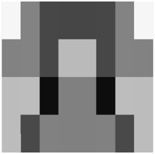
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u,v) \mathcal{B}_{uv}$.

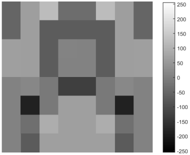
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$



$\tilde{\mathcal{I}}_5$

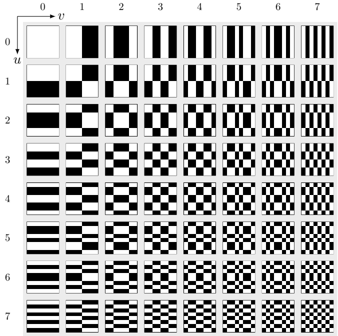


$\mathcal{I} - \tilde{\mathcal{I}}_5$

Example

... finally!

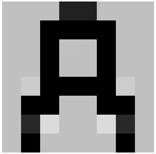
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

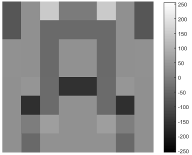
Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v) \mathcal{B}_{uv}$.

$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



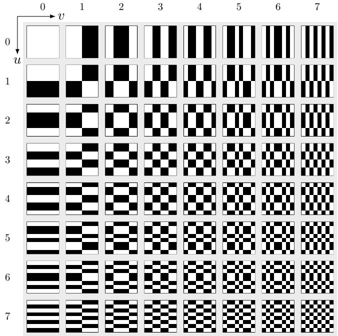
$\mathcal{I}$

$\tilde{\mathcal{I}}_6$



$\mathcal{I} - \tilde{\mathcal{I}}_6$

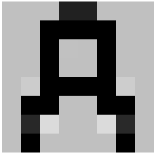
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8}\begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

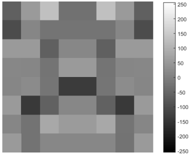
Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v)\mathcal{B}_{uv}$.

$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$

$\tilde{\mathcal{I}}_7$

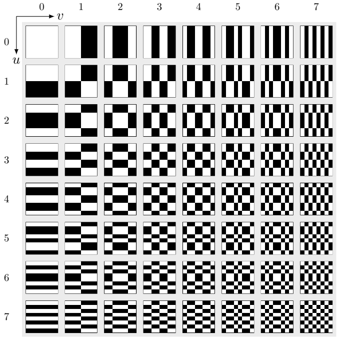


$\mathcal{I} - \tilde{\mathcal{I}}_7$

Example

... finally!

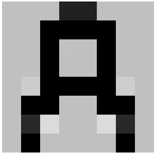
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



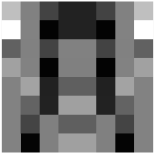
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v) \mathcal{B}_{uv}$.

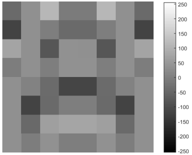
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$



$\tilde{\mathcal{I}}_8$

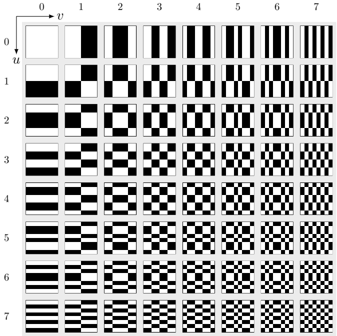


$\mathcal{I} - \tilde{\mathcal{I}}_8$

Example

... finally!

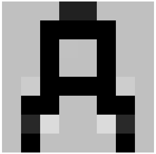
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



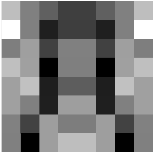
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u,v) \mathcal{B}_{uv}$.

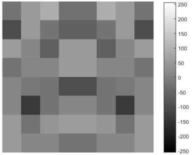
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$



$\tilde{\mathcal{I}}_9$

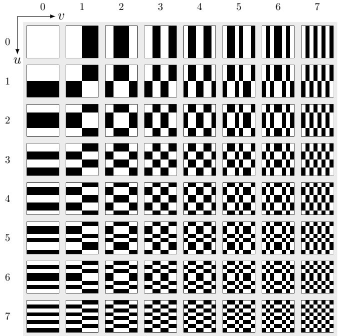


$\mathcal{I} - \tilde{\mathcal{I}}_9$

Example

... finally!

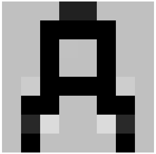
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



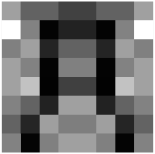
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v) \mathcal{B}_{uv}$.

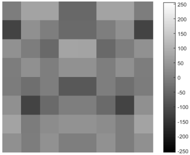
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$



$\tilde{\mathcal{I}}_{10}$

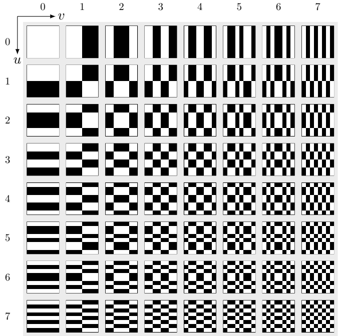


$\mathcal{I} - \tilde{\mathcal{I}}_{10}$

Example

... finally!

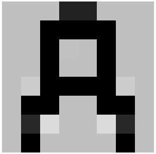
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



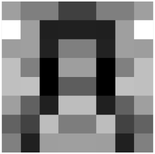
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v) \mathcal{B}_{uv}$.

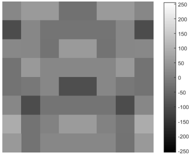
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$



$\tilde{\mathcal{I}}_{11}$

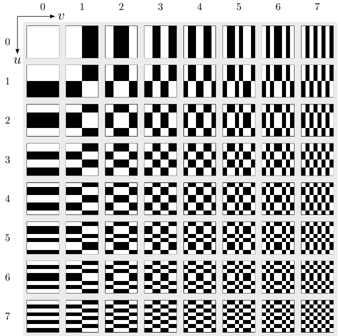


$\mathcal{I} - \tilde{\mathcal{I}}_{11}$

Example

... finally!

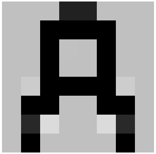
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



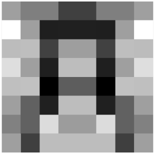
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v) \mathcal{B}_{uv}$.

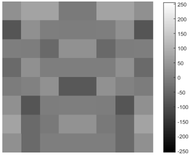
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$



$\tilde{\mathcal{I}}_{12}$

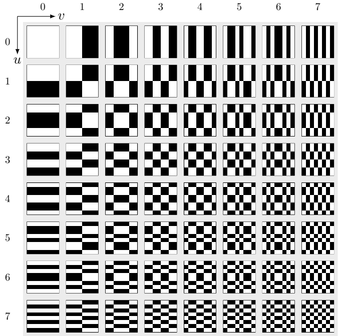


$\mathcal{I} - \tilde{\mathcal{I}}_{12}$

Example

... finally!

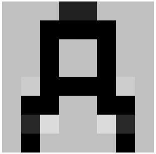
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



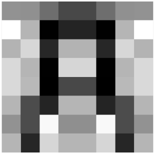
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v) \mathcal{B}_{uv}$.

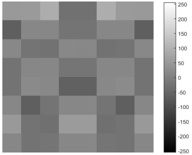
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$



$\tilde{\mathcal{I}}_{13}$

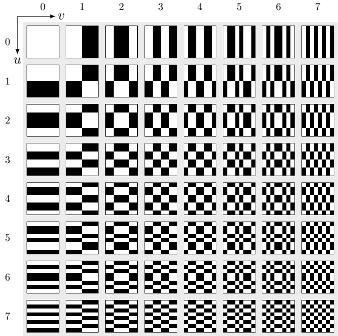


$\mathcal{I} - \tilde{\mathcal{I}}_{13}$

Example

... finally!

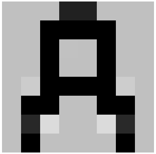
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



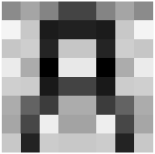
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v) \mathcal{B}_{uv}$.

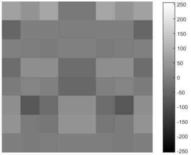
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$

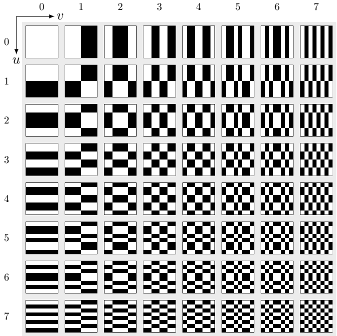


$\tilde{\mathcal{I}}_{14}$



$\mathcal{I} - \tilde{\mathcal{I}}_{14}$

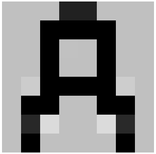
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



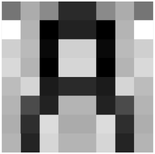
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v) \mathcal{B}_{uv}$.

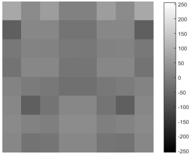
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$

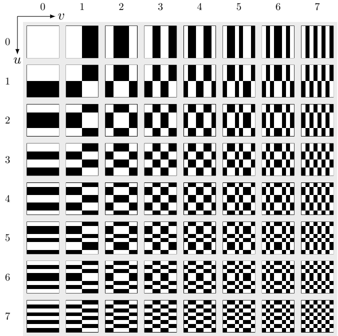


$\tilde{\mathcal{I}}_{15}$



$\mathcal{I} - \tilde{\mathcal{I}}_{15}$

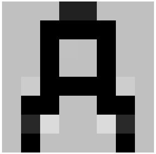
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



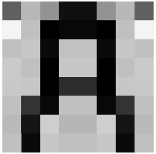
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8}\begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v)\mathcal{B}_{uv}$.

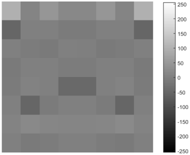
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$



$\tilde{\mathcal{I}}_{16}$

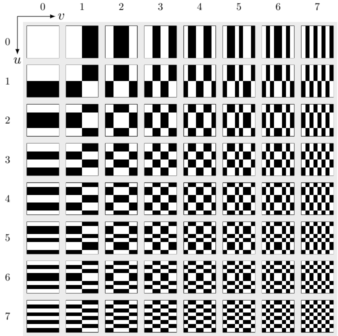


$\mathcal{I} - \tilde{\mathcal{I}}_{16}$

Example

... finally!

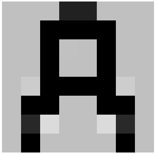
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



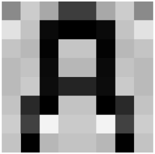
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v) \mathcal{B}_{uv}$.

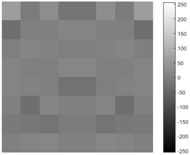
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$



$\tilde{\mathcal{I}}_{20}$

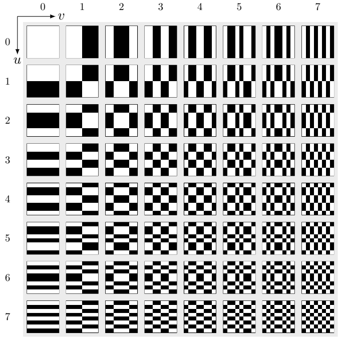


$\mathcal{I} - \tilde{\mathcal{I}}_{20}$

Example

... finally!

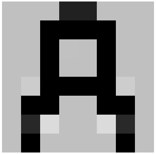
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



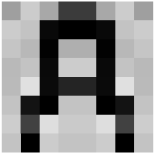
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v) \mathcal{B}_{uv}$.

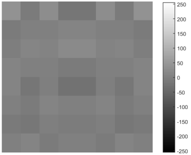
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$



$\tilde{\mathcal{I}}_{24}$

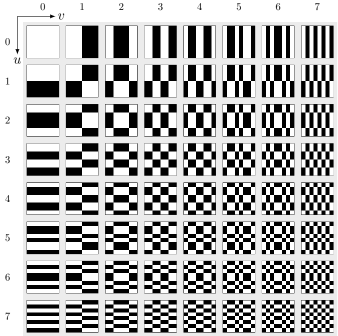


$\mathcal{I} - \tilde{\mathcal{I}}_{24}$

Example

... finally!

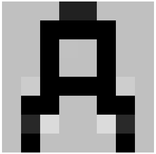
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



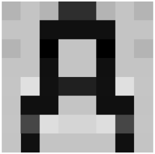
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 265 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 265 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u,v) \mathcal{B}_{uv}$.

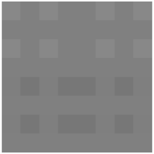
$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.

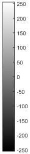


$\mathcal{I}$



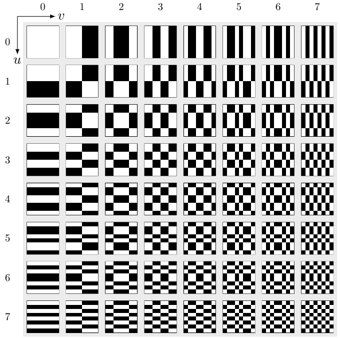
$\tilde{\mathcal{I}}_{28}$





$\mathcal{I} - \tilde{\mathcal{I}}_{28}$

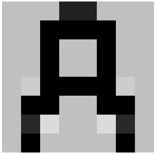
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



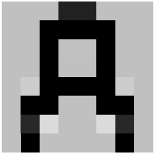
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 291 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 291 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v) \mathcal{B}_{uv}$.

$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$



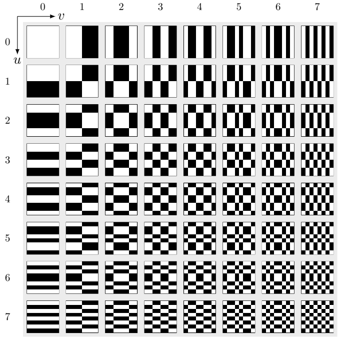
$\tilde{\mathcal{I}}_{32}$

$\mathcal{I} - \tilde{\mathcal{I}}_{32}$

Example

... finally!

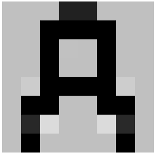
Let's take this 8×8 image $\mathcal{I} =$  and compute its Walsh-Hadamard spectrum by $\mathcal{J}_{\mathbf{H}} = \mathbf{H}_8 \mathcal{I} \mathbf{H}_8$.



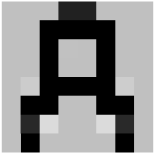
$$\mathcal{J}_{\mathbf{H}} = \frac{1}{8} \begin{bmatrix} 8291 & -1 & 1905 & -3 & 1827 & 3 & 291 & 1 \\ 291 & 3 & 1801 & 1 & -925 & -1 & -1167 & -3 \\ 559 & -1 & -1215 & 1 & -1197 & -1 & 1853 & 1 \\ -1197 & -1 & 1853 & 1 & -977 & -1 & 321 & 1 \\ 339 & -1 & 317 & 1 & -977 & -1 & 321 & 1 \\ 559 & -1 & -1215 & 1 & 339 & -1 & 317 & 1 \\ 291 & 3 & -1271 & 1 & 611 & -1 & 369 & -3 \\ 611 & -1 & 369 & -3 & -1245 & 3 & 291 & 1 \end{bmatrix}$$

Theory gives $\mathcal{I} = \sum_{u=0}^{7} \sum_{v=0}^{7} \mathcal{J}_{\mathbf{H}}(u, v) \mathcal{B}_{uv}$.

$\rightarrow$ approximate $\tilde{\mathcal{I}}$ using the leading coefficients of $\mathcal{J}_{\mathbf{H}}$.



$\mathcal{I}$

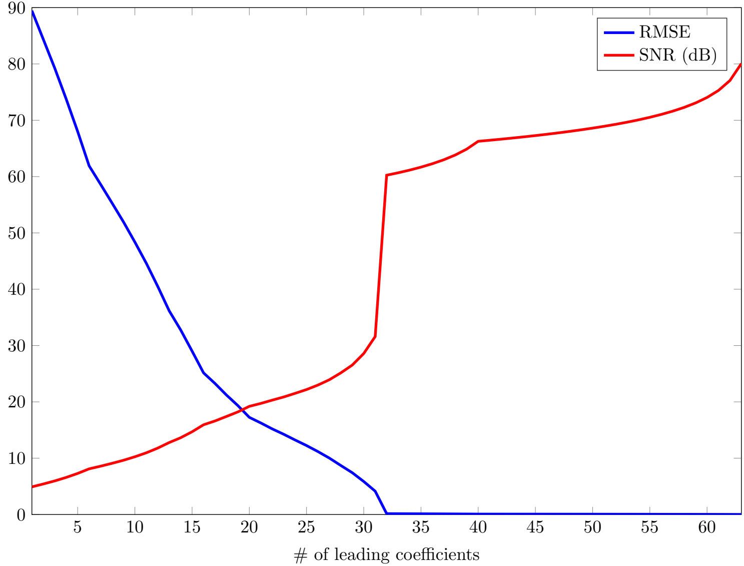


$\tilde{\mathcal{I}}_{64}$

$\mathcal{I} - \tilde{\mathcal{I}}_{64}$

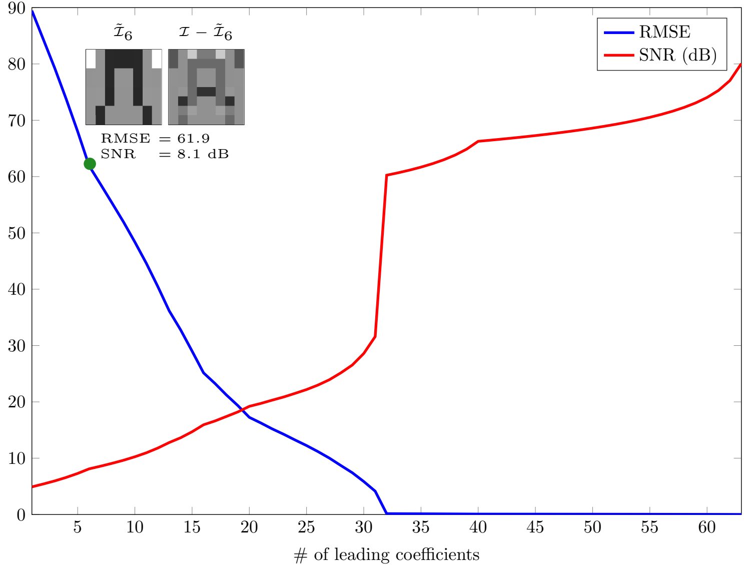
Walsh-Hadamard reconstruction error

What is really happening



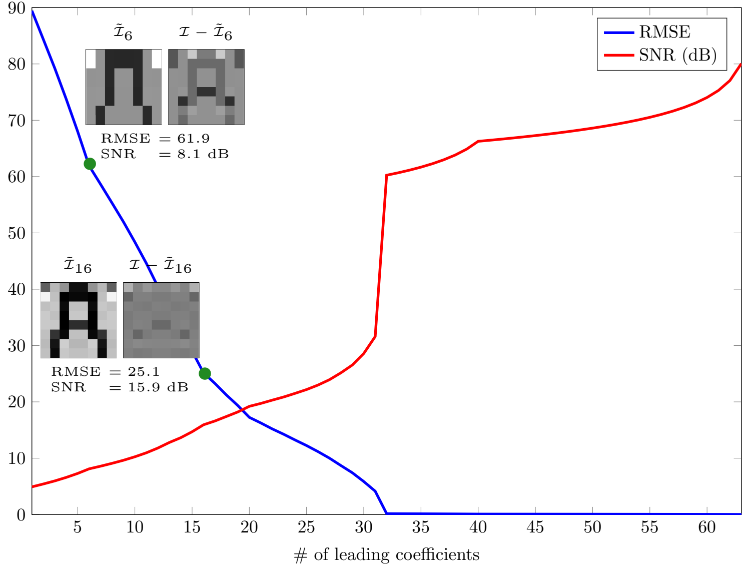
Walsh-Hadamard reconstruction error

What is really happening



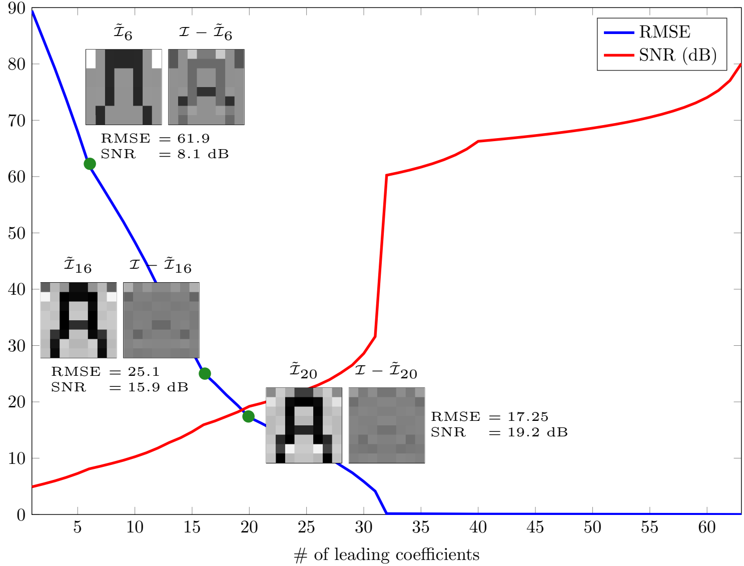
Walsh-Hadamard reconstruction error

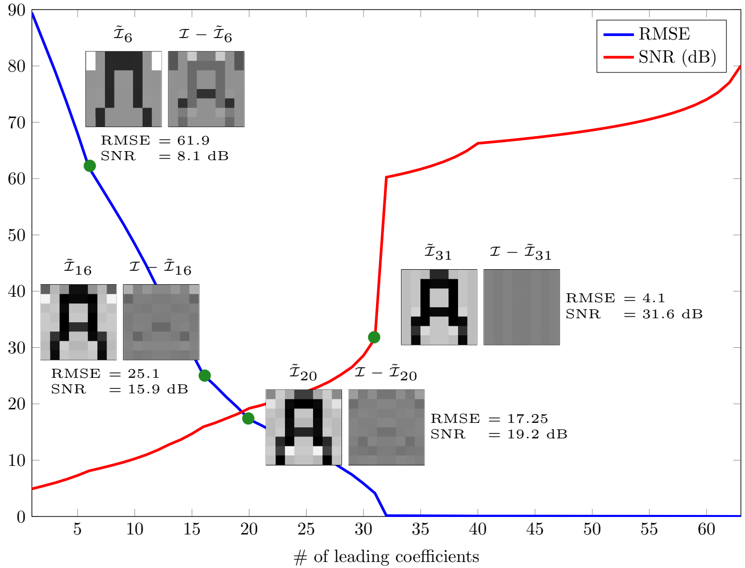
What is really happening



Walsh-Hadamard reconstruction error

What is really happening

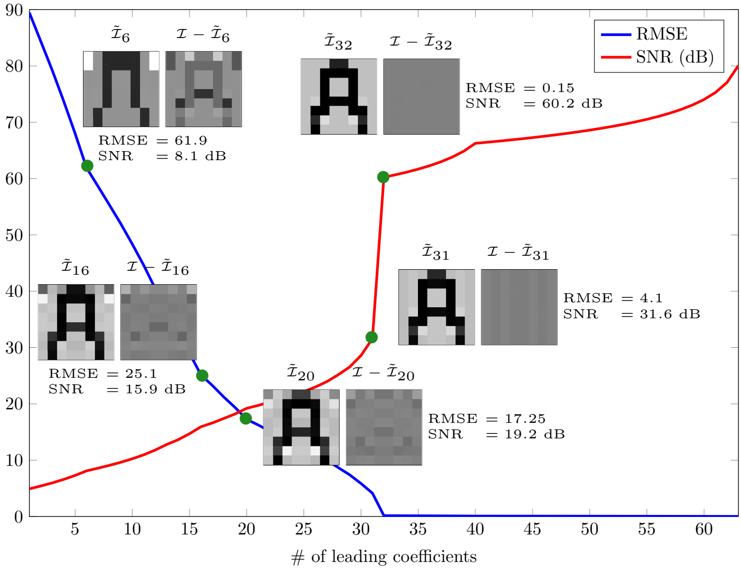




RMSE
SNR (dB)
$\tilde{\mathcal{I}}_6$
$\mathcal{I} - \tilde{\mathcal{I}}_6$
RMSE = 61.9
SNR = 8.1 dB
$\tilde{\mathcal{I}}_{16}$
$\mathcal{I} - \tilde{\mathcal{I}}_{16}$
RMSE = 25.1
SNR = 15.9 dB
$\tilde{\mathcal{I}}_{20}$
$\mathcal{I} - \tilde{\mathcal{I}}_{20}$
RMSE = 17.25
SNR = 19.2 dB
$\tilde{\mathcal{I}}_{31}$
$\mathcal{I} - \tilde{\mathcal{I}}_{31}$
RMSE = 4.1
SNR = 31.6 dB
# of leading coefficients

Walsh-Hadamard reconstruction error

What is really happening

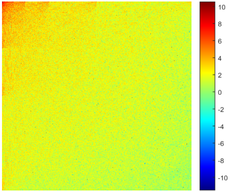


Summary on the Walsh-Hadamard transform

The Walsh-Hadamard transform was used in practice in the 60's in several space missions to compress images of:

- $\rightarrow$ the far side of the moon (*Luna 3* Soviet probe).
- $\rightarrow$ Jupiter, Saturn, Uranus, Neptune and their moons (*Mariner* and *Voyager* space probes).

512×512 image $\mathcal{I}$



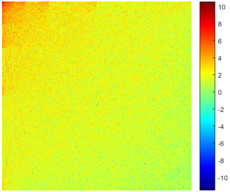
$\log(|\mathcal{J}_{\mathbf{H}}|)$

The Walsh-Hadamard transform was used in practice in the 60's in several space missions to compress images of:

→ the far side of the moon (*Luna 3* Soviet probe).

→ Jupiter, Saturn, Uranus, Neptune and their moons (*Mariner* and *Voyager* space probes).

512×512 image $\mathcal{I}$



$\log(|\mathcal{J}_{\mathbf{H}}|)$

✓ Suited to perform the frequency analysis of an image.

✓ Very fast, only requires addition and subtraction.

✗ Image dimensions must be a power of 2.

✓ Very efficient FFT-like implementation possible.

✗ Not so good to compact the image energy in a very few coefficients.

In the continuous setting.

If $f : \mathbb{R} \to \mathbb{R}, t \mapsto f(t)$ is a continuous and integrable function, and $\hat{f} : \mathbb{R} \to \mathbb{C}$, $\nu \mapsto \hat{f}(\nu)$ is its Fourier transform, then f and $\hat{f}$ are linked by:

$$\hat{f}(\nu) = \int_{-\infty}^{+\infty} f(t) e^{-i2\pi\nu t} dt \quad \text{and} \quad f(t) = \int_{-\infty}^{+\infty} \hat{f}(\nu) e^{i2\pi\nu t} d\nu$$

The discrete Fourier transform

In 1D

In the continuous setting.

If $f : \mathbb{R} \rightarrow \mathbb{R}, t \mapsto f(t)$ is a continuous and integrable function, and $\hat{f} : \mathbb{R} \rightarrow \mathbb{C}$, $\nu \mapsto \hat{f}(\nu)$ is its Fourier transform, then f and $\hat{f}$ are linked by:

$$\hat{f}(\nu) = \int_{-\infty}^{+\infty} f(t) e^{-i2\pi\nu t} dt \quad \text{and} \quad f(t) = \int_{-\infty}^{+\infty} \hat{f}(\nu) e^{i2\pi\nu t} d\nu$$

In the discrete setting.

Let $f = \{f(0), f(1), \ldots, f(M-1)\}$ be a discrete function of length M. Its *discrete* Fourier transform is the complex-valued function $\hat{f} = \{\hat{f}(0), \hat{f}(1), \ldots, \hat{f}(M-1)\}$ of length M defined as:

$$\boxed{\hat{f}(u) = \sum_{m=0}^{M-1} f(m) e^{-i2\pi\frac{mu}{M}}} \quad \text{and} \quad \boxed{f(m) = \frac{1}{M} \sum_{u=0}^{M-1} \hat{f}(u) e^{i2\pi\frac{mu}{M}}}$$

⚠ An alternative definition uses a $\frac{1}{\sqrt{M}}$ normalizing coefficient in front of the forward and inverse definitions.

The discrete Fourier transform

In 2D

Straightforward extension from 1D to 2D:

If $\mathcal{I} = \mathcal{I}(m, n)$ is a $M \times N$ image, then its 2D discrete Fourier transform $\mathcal{J}_\mathbf{F}$ is the $M \times N$ complex matrix defined as:

$$\boxed{\mathcal{J}_\mathbf{F}(u, v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \mathcal{I}(m, n) e^{-i2\pi\left(\frac{mu}{M} + \frac{nv}{N}\right)}}$$

$\rightarrow$ The forward DFT kernel is $\phi(m, n, u, v) = e^{-i2\pi\left(\frac{mu}{M} + \frac{nv}{N}\right)}$

Straightforward extension from 1D to 2D:

If $\mathcal{I} = \mathcal{I}(m,n)$ is a $M \times N$ image, then its 2D discrete Fourier transform $\mathcal{J}_{\mathbf{F}}$ is the $M \times N$ complex matrix defined as:

$$\boxed{\mathcal{J}_{\mathbf{F}}(u,v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \mathcal{I}(m,n) e^{-i2\pi\left(\frac{mu}{M} + \frac{nv}{N}\right)}}$$

$\rightarrow$ The forward DFT kernel is $\phi(m,n,u,v) = e^{-i2\pi\left(\frac{mu}{M} + \frac{nv}{N}\right)}$

The image $\mathcal{I}$ can be retrieved from $\mathcal{J}_{\mathbf{F}}$ by the 2D inverse discrete Fourier transform:

$$\boxed{\mathcal{I}(m,n) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \mathcal{J}_{\mathbf{F}}(u,v) e^{i2\pi\left(\frac{mu}{M} + \frac{nv}{N}\right)}}$$

$\rightarrow$ The inverse DFT kernel is $\psi(m,n,u,v) = \frac{1}{MN} e^{i2\pi\left(\frac{mu}{M} + \frac{nv}{N}\right)}$

Straightforward extension from 1D to 2D:

If $\mathcal{I} = \mathcal{I}(m, n)$ is a $M \times N$ image, then its 2D discrete Fourier transform $\mathcal{J}_{\mathbf{F}}$ is the $M \times N$ complex matrix defined as:

$$\boxed{\mathcal{J}_{\mathbf{F}}(u, v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \mathcal{I}(m, n) e^{-i2\pi\left(\frac{mu}{M} + \frac{nv}{N}\right)}}$$

$\rightarrow$ The forward DFT kernel is $\phi(m, n, u, v) = e^{-i2\pi\left(\frac{mu}{M} + \frac{nv}{N}\right)}$

The image $\mathcal{I}$ can be retrieved from $\mathcal{J}_{\mathbf{F}}$ by the 2D inverse discrete Fourier transform:

$$\boxed{\mathcal{I}(m, n) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \mathcal{J}_{\mathbf{F}}(u, v) e^{i2\pi\left(\frac{mu}{M} + \frac{nv}{N}\right)}}$$

$\rightarrow$ The inverse DFT kernel is $\psi(m, n, u, v) = \frac{1}{MN} e^{i2\pi\left(\frac{mu}{M} + \frac{nv}{N}\right)}$

$\Rightarrow$ Forward and inverse DFT kernels are separable.
$\Rightarrow$ $\mathcal{J}_{\mathbf{F}} = \mathbf{F}_M \mathcal{I} \mathbf{F}_N^T$ and $\mathcal{I} = \mathbf{F}_M^{-1} \mathcal{J}_{\mathbf{F}} \mathbf{F}_N^{-T}$.

The DFT kernel matrix

For a $N \times N$ image, the (j, k)-th entry of the Fourier kernel matrix $\mathbf{F}_N$ is

$\mathbf{F}_N(j, k) = e^{-i2\pi \frac{jk}{N}}$

The DFT kernel matrix

For a $N \times N$ image, the (j, k)-th entry of the Fourier kernel matrix $\mathbf{F}_N$ is
$\mathbf{F}_N(j, k) = \left(e^{-i\frac{2\pi}{N}}\right)^{jk}$

The DFT kernel matrix

For a $N \times N$ image, the (j, k)-th entry of the Fourier kernel matrix $\mathbf{F}_N$ is $\mathbf{F}_N(j, k) = \omega_N^{jk}$ with $\omega_N = e^{-i\frac{2\pi}{N}}$ is a primitive N^{th} root of unity.

The DFT kernel matrix

For a $N \times N$ image, the (j,k)-th entry of the Fourier kernel matrix $\mathbf{F}_N$ is $\mathbf{F}_N(j,k) = \omega_N^{jk}$ with $\omega_N = e^{-i\frac{2\pi}{N}}$ is a primitive N$^{\text{th}}$ root of unity.

$$\Rightarrow \mathbf{F}_N = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N & \omega_N^2 & \omega_N^3 & \dots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \omega_N^6 & \dots & \omega_N^{2(N-1)} \\ 1 & \omega_N^3 & \omega_N^6 & \omega_N^9 & \dots & \omega_N^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \omega_N^{3(N-1)} & \dots & \omega_N^{(N-1)(N-1)} \end{bmatrix}$$

The DFT kernel matrix

For a $N \times N$ image, the (j,k)-th entry of the Fourier kernel matrix $\mathbf{F}_N$ is $\mathbf{F}_N(j,k) = \omega_N^{jk}$ with $\omega_N = e^{-i\frac{2\pi}{N}}$ is a primitive N$^{\text{th}}$ root of unity.

$$\Rightarrow \mathbf{F}_N = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N & \omega_N^2 & \omega_N^3 & \dots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \omega_N^6 & \dots & \omega_N^{2(N-1)} \\ 1 & \omega_N^3 & \omega_N^6 & \omega_N^9 & \dots & \omega_N^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \omega_N^{3(N-1)} & \dots & \omega_N^{(N-1)(N-1)} \end{bmatrix}$$

Clearly, $\mathbf{F}_N = \mathbf{F}_N^T \Rightarrow$ Fourier kernel matrices are symmetric.
But it can also be shown that $\mathbf{F}_N^\dagger \mathbf{F}_N = \mathbf{F}_N \mathbf{F}_N^\dagger = N\mathbf{I}_N \Rightarrow \mathbf{F}_N^{-1} = \frac{1}{N}\mathbf{F}_N^\dagger$ with $\mathbf{F}_N^\dagger = \left(\overline{\mathbf{F}_N}\right)^T = \overline{\mathbf{F}_N^T}$ being the conjugate transpose of $\mathbf{F}_N$.

The DFT kernel matrix

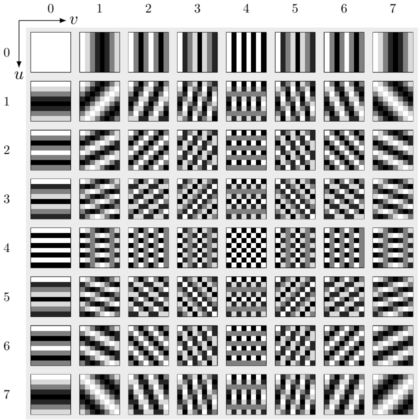
For a $N \times N$ image, the (j,k)-th entry of the Fourier kernel matrix $\mathbf{F}_N$ is $\mathbf{F}_N(j,k) = \omega_N^{jk}$ with $\omega_N = e^{-i\frac{2\pi}{N}}$ is a primitive N$^{\text{th}}$ root of unity.

$$\Rightarrow \mathbf{F}_N = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N & \omega_N^2 & \omega_N^3 & \dots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \omega_N^6 & \dots & \omega_N^{2(N-1)} \\ 1 & \omega_N^3 & \omega_N^6 & \omega_N^9 & \dots & \omega_N^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \omega_N^{3(N-1)} & \dots & \omega_N^{(N-1)(N-1)} \end{bmatrix}$$

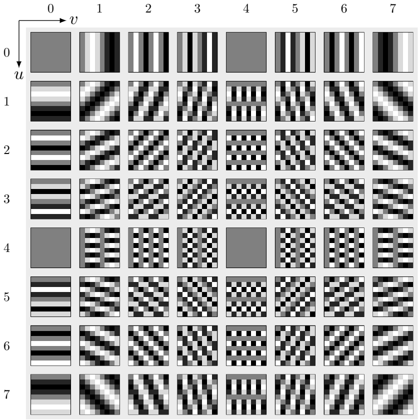
Clearly, $\mathbf{F}_N = \mathbf{F}_N^T \Rightarrow$ Fourier kernel matrices are symmetric.
But it can also be shown that $\mathbf{F}_N^\dagger \mathbf{F}_N = \mathbf{F}_N \mathbf{F}_N^\dagger = N\mathbf{I}_N \Rightarrow \mathbf{F}_N^{-1} = \frac{1}{N}\mathbf{F}_N^\dagger$ with $\mathbf{F}_N^\dagger = \left(\overline{\mathbf{F}_N}\right)^T = \overline{\mathbf{F}_N^T}$ being the conjugate transpose of $\mathbf{F}_N$.

At the end of the day, the DFT matrix $\mathcal{J}_{\mathbf{F}}$ of a $N \times N$ image $\mathcal{I}$ is given by $\mathcal{J}_{\mathbf{F}} = \mathbf{F}_N \mathcal{I} \mathbf{F}_N$ and the inverse DFT is $\mathcal{I} = \frac{1}{N^2}\mathbf{F}_N^\dagger \mathcal{J}_{\mathbf{F}} \mathbf{F}_N^\dagger$

DFT basis images for $N = 8$



$\Re\left(\mathcal{B}_{uv}\right)$



$\Im\left(\mathcal{B}_{uv}\right)$

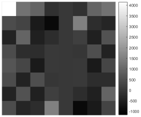
Real and imaginary parts of DFT basis images $\mathcal{B}_{uv}$ for $N = 8$. $\square = \frac{1}{64} = -\blacksquare$. The origin of each basis image is at its top-left corner.

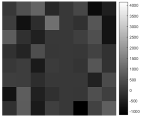
Limits of the DFT for image compression

Let's take again our 8×8 image $\mathcal{I} =$  and compute its DFT spectrum by $\mathcal{J}_{\mathbf{F}} = \mathbf{F}_8 \mathcal{I} \mathbf{F}_8^T$.

Limits of the DFT for image compression

Let's take again our 8×8 image $\mathcal{I} =$  and compute its DFT spectrum by $\mathcal{J}_{\mathbf{F}} = \mathbf{F}_8 \mathcal{I} \mathbf{F}_8^T$.

$\mathcal{J}_{\mathbf{F}} = \Big[$ 

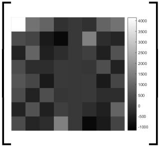
$\Big] + i \Big[$ 

$\Big]$

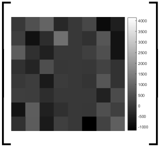
Limits of the DFT for image compression

Let's take again our 8×8 image $\mathcal{I} =$  and compute its DFT spectrum by $\mathcal{J}_{\mathbf{F}} = \mathbf{F}_8 \mathcal{I} \mathbf{F}_8^T$.

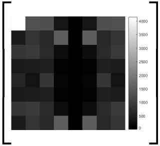
$$\mathcal{J}_{\mathbf{F}} =$$



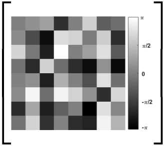
$$+\, i$$



$$=$$



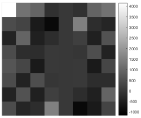
$$\times \exp \left(i \right.$$

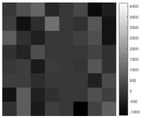


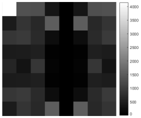
$$\left. \right)$$

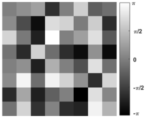
Limits of the DFT for image compression

Let's take again our 8×8 image $\mathcal{I} =$  and compute its DFT spectrum by $\mathcal{J}_{\mathbf{F}} = \mathbf{F}_8 \mathcal{I} \mathbf{F}_8^T$.

$\mathcal{J}_{\mathbf{F}} =$ 

$+\, i$ 

$=$ 

$\times \exp \left(i \right.$ 

$\left. \right)$

The DFT is widely used for general spectral analysis applications. But it suffers from two drawbacks for image compression purposes:

- Complex-valued transformed $\rightarrow$ requires double memory for storage.
- Energy compaction is not optimal (spread in both the real and imaginary coefficients of the resulting spectrum).

The discrete cosine transform

→ Published in 1974 by N. Ahmed, T. Natarajan, and K. R. Rao (Discrete cosine transform. *IEEE transactions on Computers*, vol. 100, no 1, pp. 90–93).

→ Real-valued transform, but strongly related to the DFT.
(DCT(*signal*)≡ DFT(*symmetrized signal*)).

→ Several definitions (*types*) exist, depending on the chosen boundary conditions.

Type-II DCT and its inverse transform (Type-III DCT)

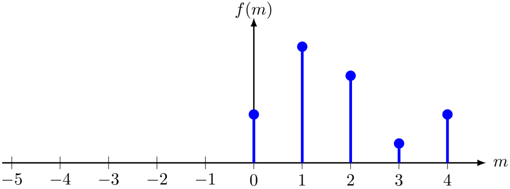
A $M \times N$ image $\mathcal{I}$ and its $M \times N$ DCT spectrum $\mathcal{J}_\mathbf{D}$ are linked by:

$$\begin{aligned}
\mathcal{J}_\mathbf{D}(u,v) &= \alpha(u)\alpha(v)\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\mathcal{I}(m,n)\cos\left(\frac{\pi(2m+1)u}{2M}\right)\cos\left(\frac{\pi(2n+1)v}{2N}\right)\\
\mathcal{I}(m,n) &= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\alpha(u)\alpha(v)\mathcal{J}_\mathbf{D}(u,v)\cos\left(\frac{\pi(2m+1)u}{2M}\right)\cos\left(\frac{\pi(2n+1)v}{2N}\right)
\end{aligned}$$

with $\alpha(u) = \begin{cases} \sqrt{\frac{1}{M}} & \text{if } u = 0 \\ \sqrt{\frac{2}{M}} & \text{if } u = 1, \dots, M-1 \end{cases}$, $\alpha(v) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } v = 0 \\ \sqrt{\frac{2}{N}} & \text{if } v = 1, \dots, N-1 \end{cases}$

Going from the DFT to the DCT

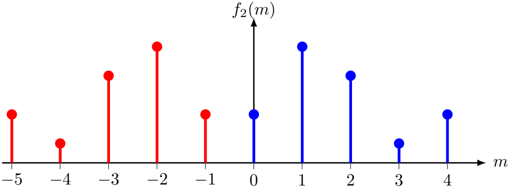
Take some M-points real signal sequence $\{f(0), \ldots, f(M-1)\}$



Going from the DFT to the DCT

Take some M-points real signal sequence $\{f(0), \ldots, f(M-1)\}$

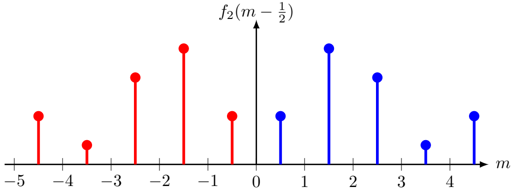
1. Create a symmetric $2M$-point sequence $f_2(m) = \begin{cases} f(m) & (0 \leq m \leq M-1) \\ f(-m-1) & (-M \leq m \leq -1) \end{cases}$.



Going from the DFT to the DCT

Take some M-points real signal sequence $\{f(0), \dots, f(M-1)\}$

1. Create a symmetric $2M$-point sequence $f_2(m) = \begin{cases} f(m) & (0 \leq m \leq M-1) \\ f(-m-1) & (-M \leq m \leq -1) \end{cases}$.
2. Right-shift the resulting sequence by $\frac{1}{2}$.

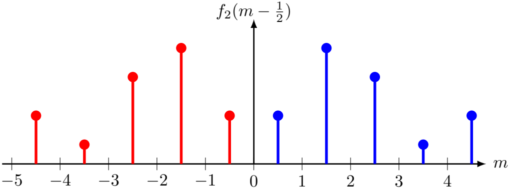


Going from the DFT to the DCT

Take some M-points real signal sequence $\{f(0), \ldots, f(M-1)\}$

1. Create a symmetric $2M$-point sequence $f_2(m) = \begin{cases} f(m) & (0 \leq m \leq M-1) \\ f(-m-1) & (-M \leq m \leq -1) \end{cases}$.
2. Right-shift the resulting sequence by $\frac{1}{2}$.

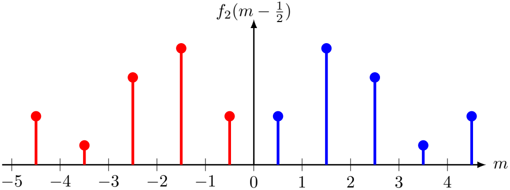
Then $\text{DCT}(f(m)) = \text{DFT}\left(f_2(m - \frac{1}{2})\right)$.



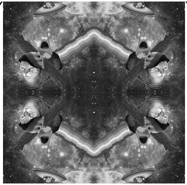
Take some M-points real signal sequence $\{f(0), \ldots, f(M-1)\}$

1. Create a symmetric $2M$-point sequence $f_2(m) = \begin{cases} f(m) & (0 \leq m \leq M-1) \\ f(-m-1) & (-M \leq m \leq -1) \end{cases}$.
2. Right-shift the resulting sequence by $\frac{1}{2}$.

Then $\mathrm{DCT}(f(m)) = \mathrm{DFT}\left(f_2(m - \frac{1}{2})\right)$.



Same goes for images:

DCT () $\equiv$ DFT ()

The DCT kernel matrix

The (j,k)-th entry of the $N \times N$ DCT matrix is $\mathbf{D}_N(j,k) = \alpha(j)\cos\left(\frac{\pi(2k+1)j}{2N}\right)$ with $\alpha(j) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } j = 0, \\ \sqrt{\frac{2}{N}} & \text{otherwise.} \end{cases}$

$$\mathbf{D}_8 = \frac{1}{2}\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \cos(\frac{\pi}{16}) & \cos(\frac{3\pi}{16}) & \cos(\frac{5\pi}{16}) & \cos(\frac{7\pi}{16}) & \cos(\frac{9\pi}{16}) & \cos(\frac{11\pi}{16}) & \cos(\frac{13\pi}{16}) & \cos(\frac{15\pi}{16}) \\ \cos(\frac{2\pi}{16}) & \cos(\frac{6\pi}{16}) & \cos(\frac{10\pi}{16}) & \cos(\frac{14\pi}{16}) & \cos(\frac{18\pi}{16}) & \cos(\frac{22\pi}{16}) & \cos(\frac{26\pi}{16}) & \cos(\frac{30\pi}{16}) \\ \cos(\frac{3\pi}{16}) & \cos(\frac{9\pi}{16}) & \cos(\frac{15\pi}{16}) & \cos(\frac{21\pi}{16}) & \cos(\frac{27\pi}{16}) & \cos(\frac{33\pi}{16}) & \cos(\frac{39\pi}{16}) & \cos(\frac{45\pi}{16}) \\ \cos(\frac{4\pi}{16}) & \cos(\frac{12\pi}{16}) & \cos(\frac{20\pi}{16}) & \cos(\frac{28\pi}{16}) & \cos(\frac{36\pi}{16}) & \cos(\frac{44\pi}{16}) & \cos(\frac{52\pi}{16}) & \cos(\frac{60\pi}{16}) \\ \cos(\frac{5\pi}{16}) & \cos(\frac{15\pi}{16}) & \cos(\frac{25\pi}{16}) & \cos(\frac{35\pi}{16}) & \cos(\frac{45\pi}{16}) & \cos(\frac{55\pi}{16}) & \cos(\frac{65\pi}{16}) & \cos(\frac{75\pi}{16}) \\ \cos(\frac{6\pi}{16}) & \cos(\frac{18\pi}{16}) & \cos(\frac{30\pi}{16}) & \cos(\frac{42\pi}{16}) & \cos(\frac{54\pi}{16}) & \cos(\frac{66\pi}{16}) & \cos(\frac{78\pi}{16}) & \cos(\frac{90\pi}{16}) \\ \cos(\frac{7\pi}{16}) & \cos(\frac{21\pi}{16}) & \cos(\frac{35\pi}{16}) & \cos(\frac{49\pi}{16}) & \cos(\frac{63\pi}{16}) & \cos(\frac{77\pi}{16}) & \cos(\frac{91\pi}{16}) & \cos(\frac{105\pi}{16}) \end{bmatrix}$$

The DCT kernel matrix

The (j,k)-th entry of the $N \times N$ DCT matrix is $\mathbf{D}_N(j,k) = \alpha(j) \cos\left(\frac{\pi(2k+1)j}{2N}\right)$ with $\alpha(j) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } j = 0, \\ \sqrt{\frac{2}{N}} & \text{otherwise.} \end{cases}$

- $\mathbf{D}_N$ is not symmetric $\rightarrow \mathcal{J}_{\mathbf{D}} = \mathbf{D}_N \mathcal{I} \mathbf{D}_N^T$
- But $\mathbf{D}_N$ is orthogonal ($\mathbf{D}_N\mathbf{D}_N^T = \mathbf{D}_N^T\mathbf{D}_N = \mathbf{I}_N$)
  $\rightarrow \mathcal{I} = \mathbf{D}_N^T \mathcal{J}_{\mathbf{D}} \mathbf{D}_N$

$$\mathbf{D}_8 = \frac{1}{2}\begin{bmatrix}
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
\cos(\frac{\pi}{16}) & \cos(\frac{3\pi}{16}) & \cos(\frac{5\pi}{16}) & \cos(\frac{7\pi}{16}) & \cos(\frac{9\pi}{16}) & \cos(\frac{11\pi}{16}) & \cos(\frac{13\pi}{16}) & \cos(\frac{15\pi}{16}) \\
\cos(\frac{2\pi}{16}) & \cos(\frac{6\pi}{16}) & \cos(\frac{10\pi}{16}) & \cos(\frac{14\pi}{16}) & \cos(\frac{18\pi}{16}) & \cos(\frac{22\pi}{16}) & \cos(\frac{26\pi}{16}) & \cos(\frac{30\pi}{16}) \\
\cos(\frac{3\pi}{16}) & \cos(\frac{9\pi}{16}) & \cos(\frac{15\pi}{16}) & \cos(\frac{21\pi}{16}) & \cos(\frac{27\pi}{16}) & \cos(\frac{33\pi}{16}) & \cos(\frac{39\pi}{16}) & \cos(\frac{45\pi}{16}) \\
\cos(\frac{4\pi}{16}) & \cos(\frac{12\pi}{16}) & \cos(\frac{20\pi}{16}) & \cos(\frac{28\pi}{16}) & \cos(\frac{36\pi}{16}) & \cos(\frac{44\pi}{16}) & \cos(\frac{52\pi}{16}) & \cos(\frac{60\pi}{16}) \\
\cos(\frac{5\pi}{16}) & \cos(\frac{15\pi}{16}) & \cos(\frac{25\pi}{16}) & \cos(\frac{35\pi}{16}) & \cos(\frac{45\pi}{16}) & \cos(\frac{55\pi}{16}) & \cos(\frac{65\pi}{16}) & \cos(\frac{75\pi}{16}) \\
\cos(\frac{6\pi}{16}) & \cos(\frac{18\pi}{16}) & \cos(\frac{30\pi}{16}) & \cos(\frac{42\pi}{16}) & \cos(\frac{54\pi}{16}) & \cos(\frac{66\pi}{16}) & \cos(\frac{78\pi}{16}) & \cos(\frac{90\pi}{16}) \\
\cos(\frac{7\pi}{16}) & \cos(\frac{21\pi}{16}) & \cos(\frac{35\pi}{16}) & \cos(\frac{49\pi}{16}) & \cos(\frac{63\pi}{16}) & \cos(\frac{77\pi}{16}) & \cos(\frac{91\pi}{16}) & \cos(\frac{105\pi}{16})
\end{bmatrix}$$

The DCT kernel matrix

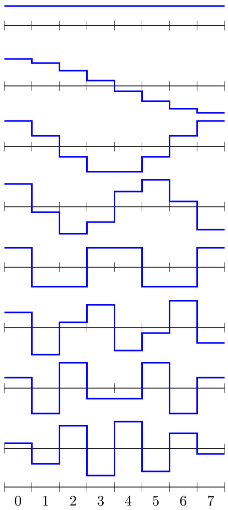
The (j,k)-th entry of the $N \times N$ DCT matrix is $\mathbf{D}_N(j,k) = \alpha(j)\cos\left(\frac{\pi(2k+1)j}{2N}\right)$

with $\alpha(j) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } j = 0, \\ \sqrt{\frac{2}{N}} & \text{otherwise.} \end{cases}$

- $\mathbf{D}_N$ is not symmetric $\rightarrow \mathcal{J}_{\mathbf{D}} = \mathbf{D}_N \mathcal{I} \mathbf{D}_N^T$
- But $\mathbf{D}_N$ is orthogonal ($\mathbf{D}_N \mathbf{D}_N^T = \mathbf{D}_N^T \mathbf{D}_N = \mathbf{I}_N$)
  $\rightarrow \mathcal{I} = \mathbf{D}_N^T \mathcal{J}_{\mathbf{D}} \mathbf{D}_N$

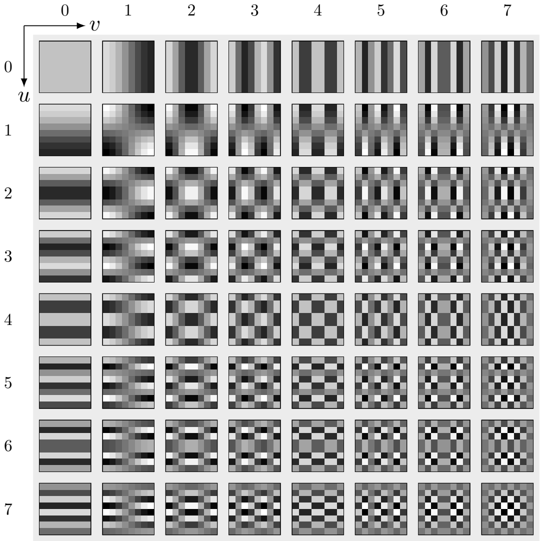
$$\mathbf{D}_8 = \frac{1}{2}\begin{bmatrix}
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
\cos(\frac{\pi}{16}) & \cos(\frac{3\pi}{16}) & \cos(\frac{5\pi}{16}) & \cos(\frac{7\pi}{16}) & \cos(\frac{9\pi}{16}) & \cos(\frac{11\pi}{16}) & \cos(\frac{13\pi}{16}) & \cos(\frac{15\pi}{16}) \\
\cos(\frac{2\pi}{16}) & \cos(\frac{6\pi}{16}) & \cos(\frac{10\pi}{16}) & \cos(\frac{14\pi}{16}) & \cos(\frac{18\pi}{16}) & \cos(\frac{22\pi}{16}) & \cos(\frac{26\pi}{16}) & \cos(\frac{30\pi}{16}) \\
\cos(\frac{3\pi}{16}) & \cos(\frac{9\pi}{16}) & \cos(\frac{15\pi}{16}) & \cos(\frac{21\pi}{16}) & \cos(\frac{27\pi}{16}) & \cos(\frac{33\pi}{16}) & \cos(\frac{39\pi}{16}) & \cos(\frac{45\pi}{16}) \\
\cos(\frac{4\pi}{16}) & \cos(\frac{12\pi}{16}) & \cos(\frac{20\pi}{16}) & \cos(\frac{28\pi}{16}) & \cos(\frac{36\pi}{16}) & \cos(\frac{44\pi}{16}) & \cos(\frac{52\pi}{16}) & \cos(\frac{60\pi}{16}) \\
\cos(\frac{5\pi}{16}) & \cos(\frac{15\pi}{16}) & \cos(\frac{25\pi}{16}) & \cos(\frac{35\pi}{16}) & \cos(\frac{45\pi}{16}) & \cos(\frac{55\pi}{16}) & \cos(\frac{65\pi}{16}) & \cos(\frac{75\pi}{16}) \\
\cos(\frac{6\pi}{16}) & \cos(\frac{18\pi}{16}) & \cos(\frac{30\pi}{16}) & \cos(\frac{42\pi}{16}) & \cos(\frac{54\pi}{16}) & \cos(\frac{66\pi}{16}) & \cos(\frac{78\pi}{16}) & \cos(\frac{90\pi}{16}) \\
\cos(\frac{7\pi}{16}) & \cos(\frac{21\pi}{16}) & \cos(\frac{35\pi}{16}) & \cos(\frac{49\pi}{16}) & \cos(\frac{63\pi}{16}) & \cos(\frac{77\pi}{16}) & \cos(\frac{91\pi}{16}) & \cos(\frac{105\pi}{16})
\end{bmatrix}$$

DCT $\equiv$ decomposition over the set of DCT basis functions.



DCT basis functions

DCT basis images for $M = N = 8$



DCT basis images $\mathcal{B}_{uv}$ for $M = N = 8$. $\square = \frac{1}{4}\cos(\frac{\pi}{16})^2$, and $\blacksquare = -\square$. The origin of each basis image is at its top-left corner.

Example

New bottle, same old wine

Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.

Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.

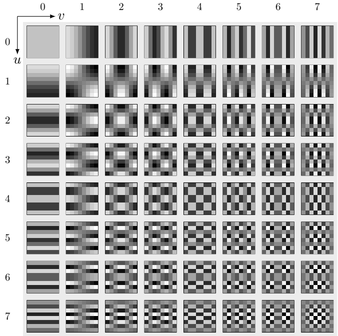
$$\Rightarrow \mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$

Example

New bottle, same old wine

Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.

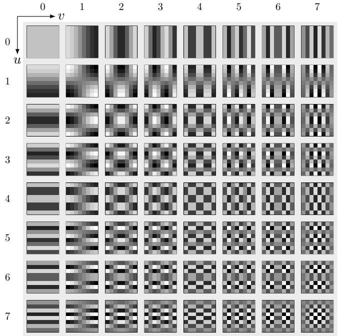
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



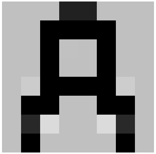
Example

New bottle, same old wine

Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.

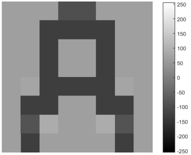


$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$

$\tilde{\mathcal{I}}_1$

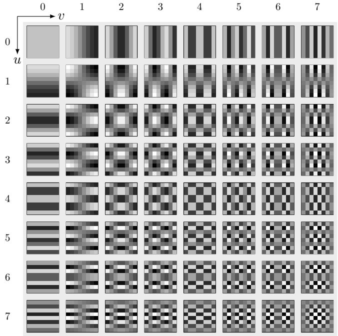


$\mathcal{I} - \tilde{\mathcal{I}}_1$

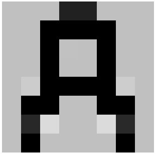
Example

New bottle, same old wine

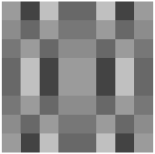
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



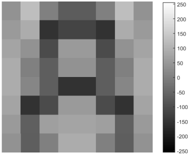
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_2$

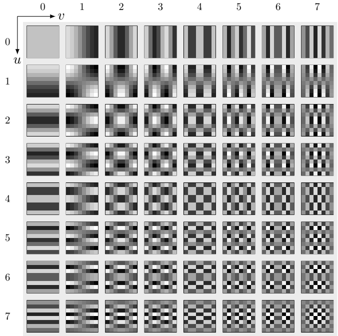


$\mathcal{I} - \tilde{\mathcal{I}}_2$

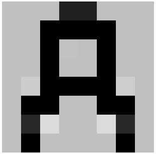
Example

New bottle, same old wine

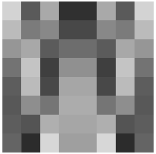
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



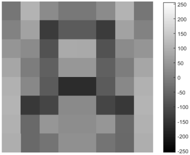
$$
\mathcal{J}_{\mathbf{D}} = \begin{bmatrix}
1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\
-14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\
78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\
-74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\
42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\
162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\
6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\
47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01
\end{bmatrix}
$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_3$

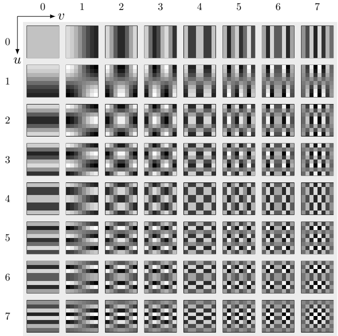


$\mathcal{I} - \tilde{\mathcal{I}}_3$

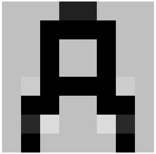
Example

New bottle, same old wine

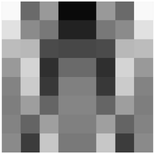
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



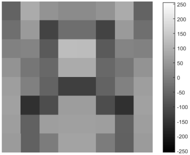
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_4$

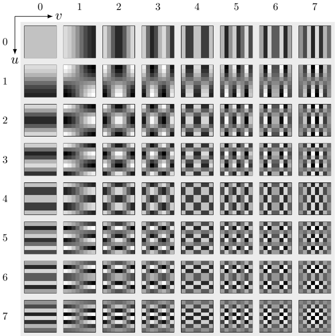


$\mathcal{I} - \tilde{\mathcal{I}}_4$

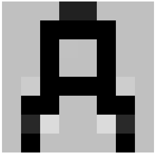
Example

New bottle, same old wine

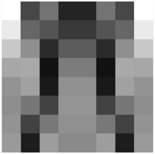
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



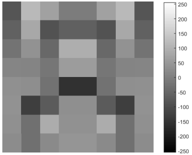
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_5$

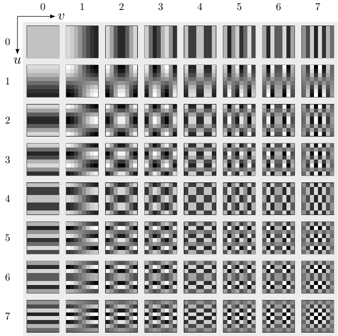


$\mathcal{I} - \tilde{\mathcal{I}}_5$

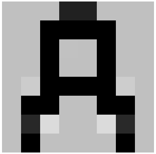
Example

New bottle, same old wine

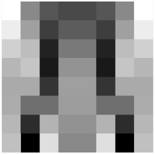
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



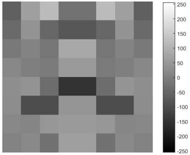
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_6$

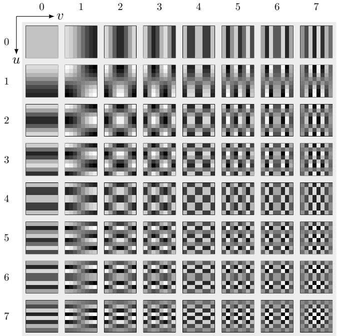


$\mathcal{I} - \tilde{\mathcal{I}}_6$

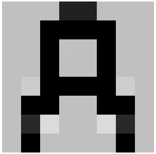
Example

New bottle, same old wine

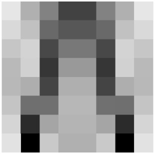
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



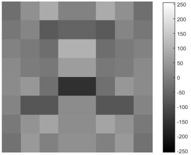
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_7$

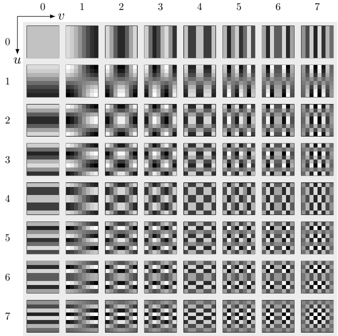


$\mathcal{I} - \tilde{\mathcal{I}}_7$

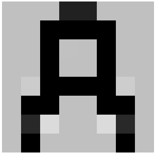
Example

New bottle, same old wine

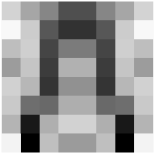
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



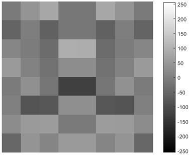
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_8$

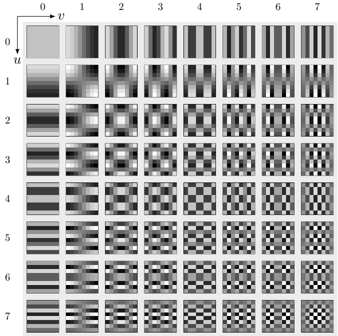


$\mathcal{I} - \tilde{\mathcal{I}}_8$

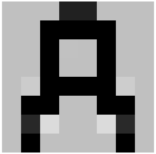
Example

New bottle, same old wine

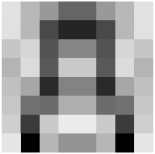
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



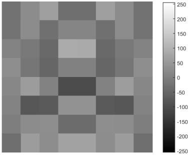
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_9$

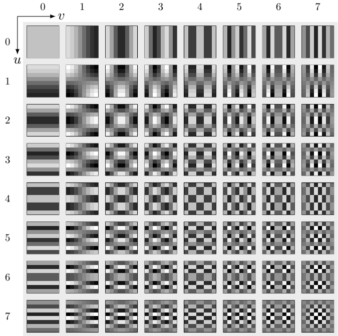


$\mathcal{I} - \tilde{\mathcal{I}}_9$

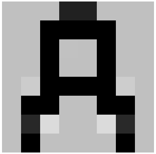
Example

New bottle, same old wine

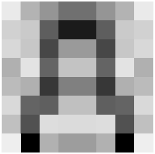
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



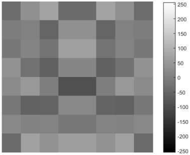
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_{10}$

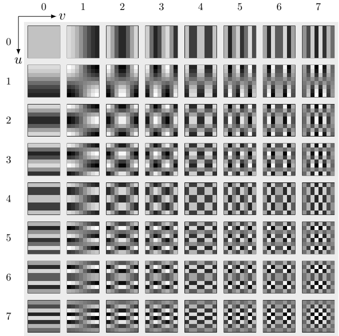


$\mathcal{I} - \tilde{\mathcal{I}}_{10}$

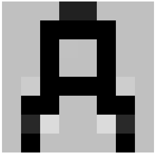
Example

New bottle, same old wine

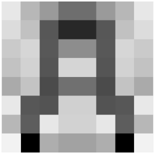
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



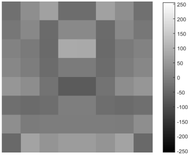
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_{11}$

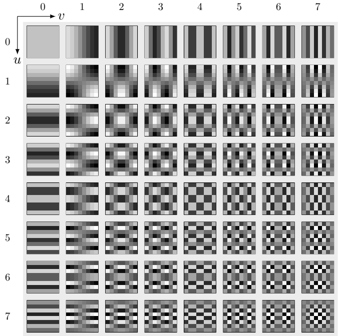


$\mathcal{I} - \tilde{\mathcal{I}}_{11}$

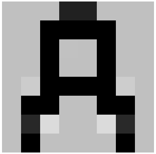
Example

New bottle, same old wine

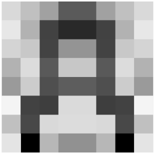
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



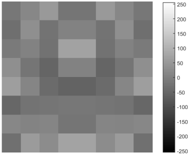
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$

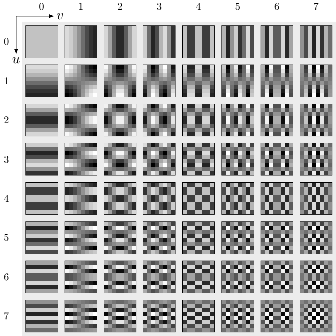


$\tilde{\mathcal{I}}_{12}$

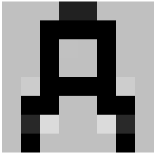


$\mathcal{I} - \tilde{\mathcal{I}}_{12}$

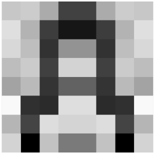
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



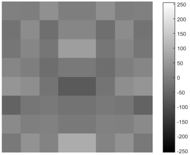
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_{13}$

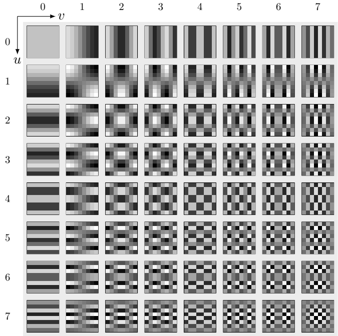


$\mathcal{I} - \tilde{\mathcal{I}}_{13}$

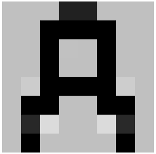
Example

New bottle, same old wine

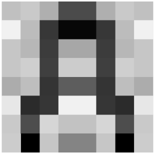
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



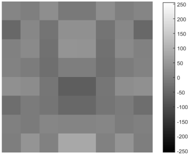
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_{14}$

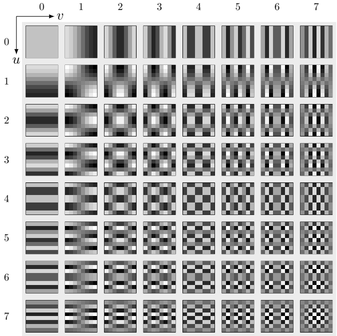


$\mathcal{I} - \tilde{\mathcal{I}}_{14}$

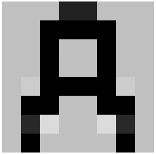
Example

New bottle, same old wine

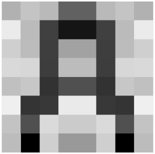
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



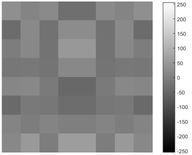
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_{15}$

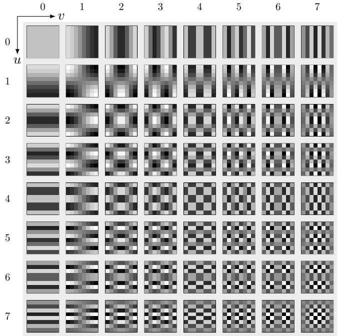


$\mathcal{I} - \tilde{\mathcal{I}}_{15}$

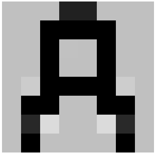
Example

New bottle, same old wine

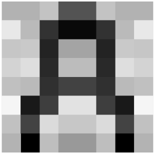
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



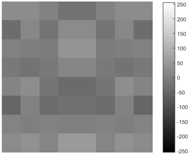
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_{16}$

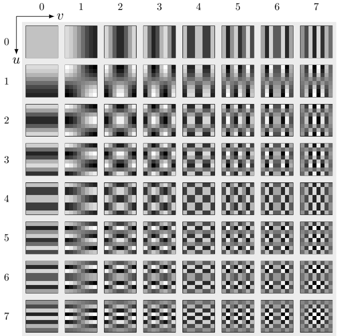


$\mathcal{I} - \tilde{\mathcal{I}}_{16}$

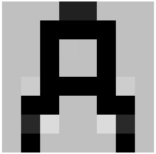
Example

New bottle, same old wine

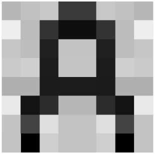
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



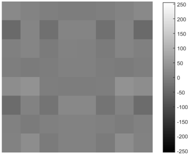
$$
\mathcal{J}_{\mathbf{D}} = \begin{bmatrix}
1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\
-14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\
78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\
-74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\
42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\
162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\
6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\
47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01
\end{bmatrix}
$$



$\mathcal{I}$

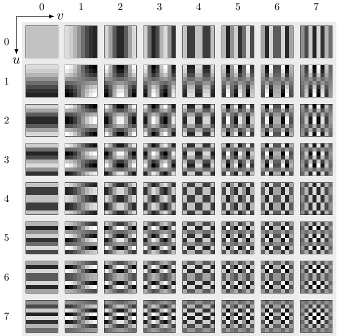


$\tilde{\mathcal{I}}_{20}$

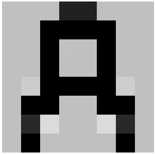


$\mathcal{I} - \tilde{\mathcal{I}}_{20}$

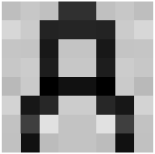
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



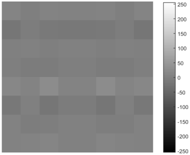
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_{24}$

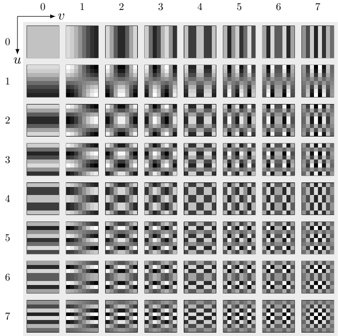


$\mathcal{I} - \tilde{\mathcal{I}}_{24}$

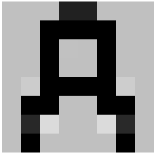
Example

New bottle, same old wine

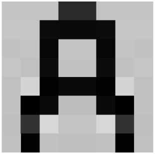
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



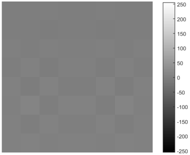
$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_{28}$

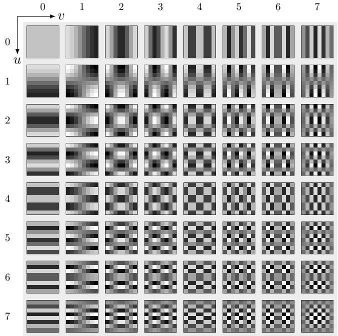


$\mathcal{I} - \tilde{\mathcal{I}}_{28}$

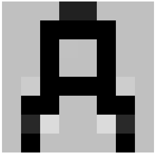
Example

New bottle, same old wine

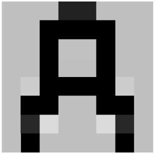
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



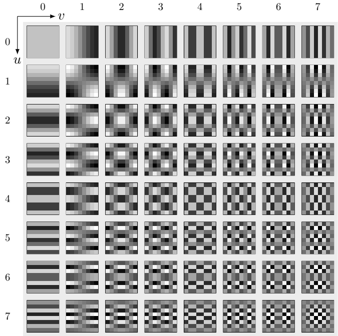
$\tilde{\mathcal{I}}_{32}$

$\mathcal{I} - \tilde{\mathcal{I}}_{32}$

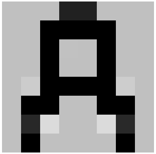
Example

New bottle, same old wine

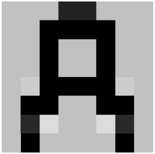
Let's take again our 8×8 image $\mathcal{I} =$  and compute its DCT spectrum by $\mathcal{J}_{\mathbf{D}} = \mathbf{D}_8 \mathcal{I} \mathbf{D}_8^T$.



$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$



$\mathcal{I}$



$\tilde{\mathcal{I}}_{64}$

$\mathcal{I} - \tilde{\mathcal{I}}_{64}$

Rounding the DCT spectrum

A first step toward lossy compression

The (j,k)-th entry of the $N \times N$ DCT matrix is $\mathbf{D}_N(j,k) = \alpha(j)\cos\left(\frac{\pi(2k+1)j}{2N}\right)$ with $\alpha(j) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } j = 0, \\ \sqrt{\frac{2}{N}} & \text{otherwise.} \end{cases}$

$$\mathbf{D}_8 = \frac{1}{2}\begin{bmatrix}
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
\cos(\frac{\pi}{16}) & \cos(\frac{3\pi}{16}) & \cos(\frac{5\pi}{16}) & \cos(\frac{7\pi}{16}) & \cos(\frac{9\pi}{16}) & \cos(\frac{11\pi}{16}) & \cos(\frac{13\pi}{16}) & \cos(\frac{15\pi}{16}) \\
\cos(\frac{2\pi}{16}) & \cos(\frac{6\pi}{16}) & \cos(\frac{10\pi}{16}) & \cos(\frac{14\pi}{16}) & \cos(\frac{18\pi}{16}) & \cos(\frac{22\pi}{16}) & \cos(\frac{26\pi}{16}) & \cos(\frac{30\pi}{16}) \\
\cos(\frac{3\pi}{16}) & \cos(\frac{9\pi}{16}) & \cos(\frac{15\pi}{16}) & \cos(\frac{21\pi}{16}) & \cos(\frac{27\pi}{16}) & \cos(\frac{33\pi}{16}) & \cos(\frac{39\pi}{16}) & \cos(\frac{45\pi}{16}) \\
\cos(\frac{4\pi}{16}) & \cos(\frac{12\pi}{16}) & \cos(\frac{20\pi}{16}) & \cos(\frac{28\pi}{16}) & \cos(\frac{36\pi}{16}) & \cos(\frac{44\pi}{16}) & \cos(\frac{52\pi}{16}) & \cos(\frac{60\pi}{16}) \\
\cos(\frac{5\pi}{16}) & \cos(\frac{15\pi}{16}) & \cos(\frac{25\pi}{16}) & \cos(\frac{35\pi}{16}) & \cos(\frac{45\pi}{16}) & \cos(\frac{55\pi}{16}) & \cos(\frac{65\pi}{16}) & \cos(\frac{75\pi}{16}) \\
\cos(\frac{6\pi}{16}) & \cos(\frac{18\pi}{16}) & \cos(\frac{30\pi}{16}) & \cos(\frac{42\pi}{16}) & \cos(\frac{54\pi}{16}) & \cos(\frac{66\pi}{16}) & \cos(\frac{78\pi}{16}) & \cos(\frac{90\pi}{16}) \\
\cos(\frac{7\pi}{16}) & \cos(\frac{21\pi}{16}) & \cos(\frac{35\pi}{16}) & \cos(\frac{49\pi}{16}) & \cos(\frac{63\pi}{16}) & \cos(\frac{77\pi}{16}) & \cos(\frac{91\pi}{16}) & \cos(\frac{105\pi}{16})
\end{bmatrix}$$

Rounding the DCT spectrum

A first step toward lossy compression

The (j,k)-th entry of the $N \times N$ DCT matrix is $\mathbf{D}_N(j,k) = \alpha(j)\cos\left(\frac{\pi(2k+1)j}{2N}\right)$ with $\alpha(j) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } j = 0, \\ \sqrt{\frac{2}{N}} & \text{otherwise.} \end{cases}$

$\rightarrow$ Entries of $\mathbf{D}_N$ are floating point numbers.

$$\mathbf{D}_8 = \frac{1}{2}\begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix}$$

Rounding the DCT spectrum

A first step toward lossy compression

The (j,k)-th entry of the $N \times N$ DCT matrix is $\mathbf{D}_N(j,k) = \alpha(j)\cos\left(\frac{\pi(2k+1)j}{2N}\right)$ with $\alpha(j) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } j = 0, \\ \sqrt{\frac{2}{N}} & \text{otherwise.} \end{cases}$

→ Entries of $\mathbf{D}_N$ are floating point numbers.

→ So are those of $\mathcal{J}_{\mathbf{D}}$.

$$\mathcal{J}_{\mathbf{D}} = \begin{bmatrix} 1036.38 & -0.26 & 232.68 & -0.03 & 228.38 & 0.49 & -60.52 & 0.02 \\ -14.65 & 0.24 & 243.27 & -0.13 & -181.82 & -0.14 & -224.28 & -0.28 \\ 78.48 & 0.09 & -97.16 & 0.03 & -109.01 & -0.21 & 290.97 & 0.02 \\ -74.41 & -0.20 & 75.28 & 0.17 & -68.36 & -0.02 & 82.06 & 0.34 \\ 42.38 & -0.03 & 51.96 & 0.10 & -122.13 & -0.15 & 21.91 & 0.17 \\ 162.51 & -0.02 & -163.91 & -0.02 & 21.13 & 0.08 & 56.39 & -0.03 \\ 6.87 & 0.32 & -99.53 & -0.19 & 127.82 & -0.13 & -8.59 & -0.41 \\ 47.59 & -0.28 & 52.52 & -0.04 & -126.96 & 0.55 & 19.85 & 0.01 \end{bmatrix}$$

Rounding the DCT spectrum

A first step toward lossy compression

The (j,k)-th entry of the $N \times N$ DCT matrix is $\mathbf{D}_N(j,k) = \alpha(j)\cos\left(\frac{\pi(2k+1)j}{2N}\right)$ with $\alpha(j) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } j = 0, \\ \sqrt{\frac{2}{N}} & \text{otherwise.} \end{cases}$

- $\rightarrow$ Entries of $\mathbf{D}_N$ are floating point numbers.
- $\rightarrow$ So are those of $\mathcal{J}_{\mathbf{D}}$.
- $\rightarrow$ Round the spectrum values $\tilde{\mathcal{J}}_{\mathbf{D}} = \lfloor \mathcal{J}_{\mathbf{D}} \rceil$

$$\tilde{\mathcal{J}}_{\mathbf{D}} = \begin{bmatrix} 1036 & 0 & 233 & 0 & 228 & 0 & -61 & 0 \\ -15 & 0 & 243 & 0 & -182 & 0 & -224 & 0 \\ 78 & 0 & -97 & 0 & -109 & 0 & 291 & 0 \\ -74 & 0 & 75 & 0 & -68 & 0 & 82 & 0 \\ 42 & 0 & 52 & 0 & -122 & 0 & 22 & 0 \\ 163 & 0 & -164 & 0 & 21 & 0 & 56 & 0 \\ 7 & 0 & -100 & 0 & 128 & 0 & -9 & 0 \\ 48 & 0 & 53 & 0 & -127 & 1 & 20 & 0 \end{bmatrix}$$

Rounding the DCT spectrum

A first step toward lossy compression

The (j, k)-th entry of the $N \times N$ DCT matrix is $\mathbf{D}_N(j,k) = \alpha(j) \cos\left(\frac{\pi(2k+1)j}{2N}\right)$ with

$$\alpha(j) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } j = 0, \\ \sqrt{\frac{2}{N}} & \text{otherwise.} \end{cases}$$

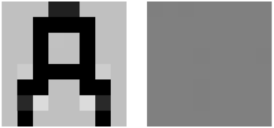
→ Entries of $\mathbf{D}_N$ are floating point numbers.

→ So are those of $\mathcal{J}_{\mathbf{D}}$.

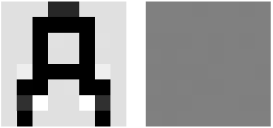
→ Round the spectrum values $\tilde{\mathcal{J}}_{\mathbf{D}} = \lfloor \mathcal{J}_{\mathbf{D}} \rceil$

$$\tilde{\mathcal{J}}_{\mathbf{D}} = \begin{bmatrix} 1036 & 0 & 233 & 0 & 228 & 0 & -61 & 0 \\ -15 & 0 & 243 & 0 & -182 & 0 & -224 & 0 \\ 78 & 0 & -97 & 0 & -109 & 0 & 291 & 0 \\ -74 & 0 & 75 & 0 & -68 & 0 & 82 & 0 \\ 42 & 0 & 52 & 0 & -122 & 0 & 22 & 0 \\ 163 & 0 & -164 & 0 & 21 & 0 & 56 & 0 \\ 7 & 0 & -100 & 0 & 128 & 0 & -9 & 0 \\ 48 & 0 & 53 & 0 & -127 & 1 & 20 & 0 \end{bmatrix}$$

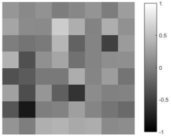
Reconstruction and error:



Without rounding

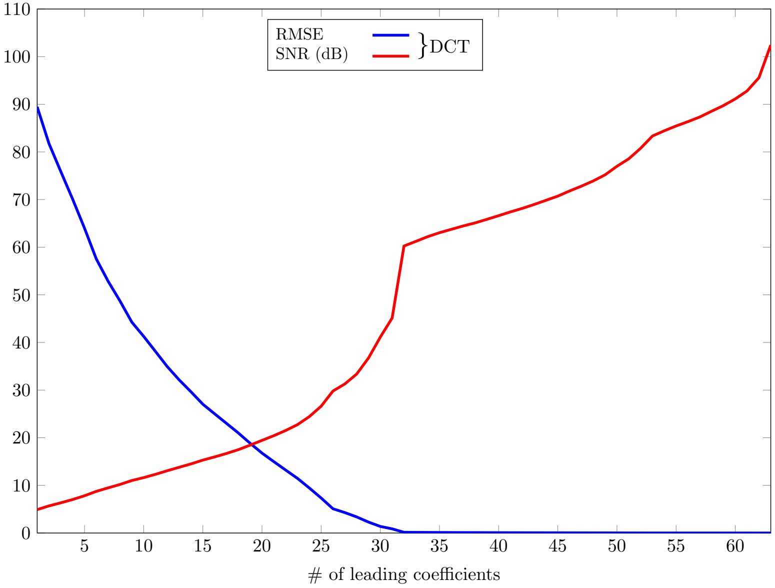


With rounding

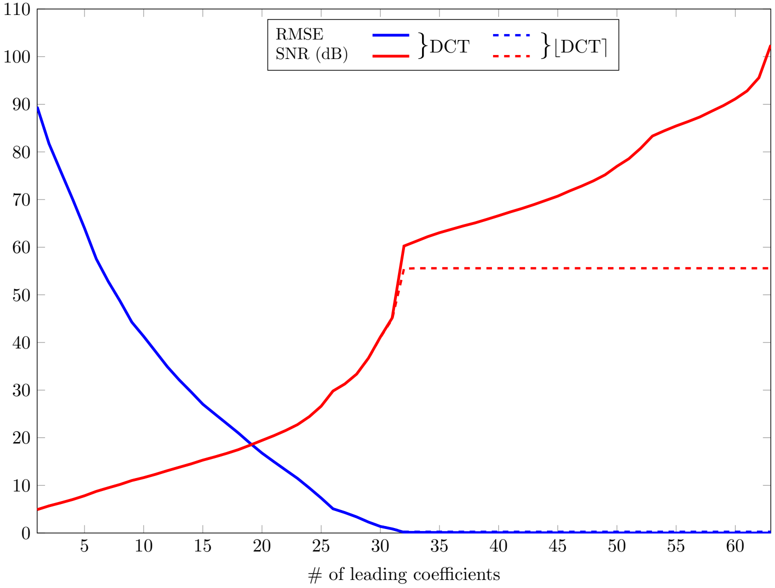


Difference

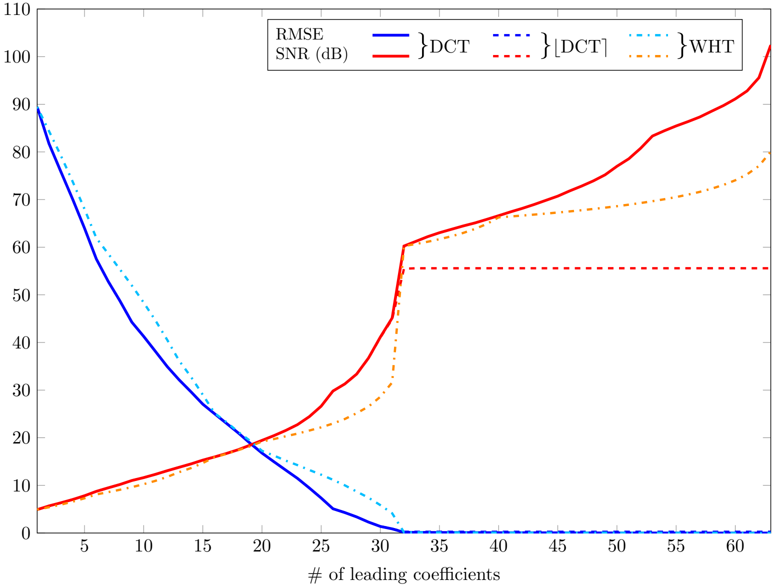
DCT Reconstruction error



DCT Reconstruction error

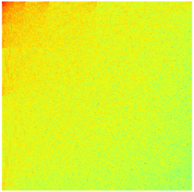


DCT Reconstruction error

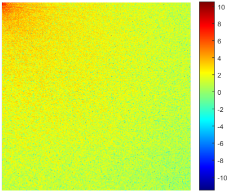


DCT vs. Walsh-Hadamard transform

512 × 512 image $\mathcal{I}$



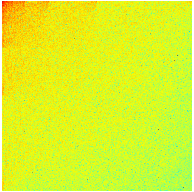
$\log(|\mathcal{J}_{\mathbf{H}}|)$



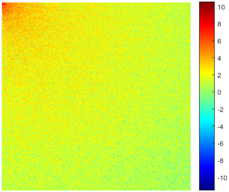
$\log(|\mathcal{J}_{\mathbf{D}}|)$

DCT vs. Walsh-Hadamard transform

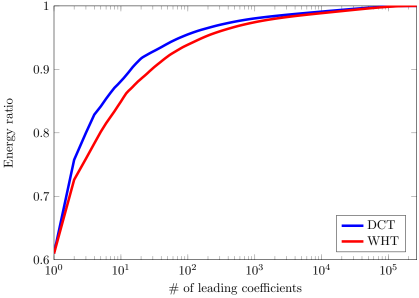
512×512 image $\mathcal{I}$



$\log(|\mathcal{J}_{\mathbf{H}}|)$



$\log(|\mathcal{J}_{\mathbf{D}}|)$



Parseval theorem gives $\|\mathcal{I}\|_2^2 = \|\mathcal{J}_{\mathbf{H}}\|_2^2 = \|\mathcal{J}_{\mathbf{D}}\|_2^2$

$\rightarrow$ A efficient transform should compact most of the image energy into as few coefficients as possible.

	80%	90%	95%	99%
DCT	4	15	77	7784
WHT	5	29	169	13501

Number of leading coefficients necessary to reach a given fraction of the total energy.

	WHT	DFT	DCT
Suited for frequency analysis	✓	✓	✓
Real result	✓	✗	✓
Computationally cheap	✓	✗	✗
Adapated to any image dimensions	✗	✓	✓
Efficient implementation available	✓	✓	✓
Energy compaction	✗	✗	✓

	WHT	DFT	DCT
Suited for frequency analysis	✓	✓	✓
Real result	✓	✗	✓
Computationally cheap	✓	✗	✗
Adapated to any image dimensions	✗	✓	✓
Efficient implementation available	✓	✓	✓
Energy compaction	✗	✗	✓

JPEG compression algorithm

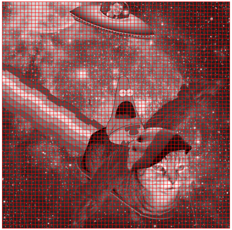
A brief overview

JPEG $\equiv$ Joint Photographic Expert Group.

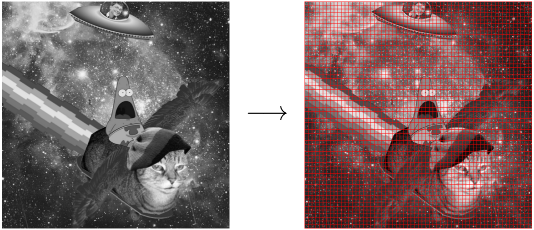
- $\rightarrow$ Started in 1986.
- $\rightarrow$ First standard (Part 1) released in 1992.
  - $\hookrightarrow$ Its pet name is ISO/CEI 10918-1 = UIT-T Recommendation T.81.
- $\rightarrow$ Latest one (Part 6) released in 2013.
- $\rightarrow$ Still active today (2 or 3 meetings per year).
- $\rightarrow$ Has spawn many compression standards (JPEG2000, JPEG XR, incorporated in MPEG . . .).

1[st] step: the input image is divided into non-overlapping 8×8 macro-blocks.

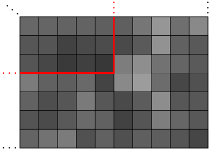
$\longrightarrow$



1st step: the input image is divided into non-overlapping 8×8 macro-blocks.

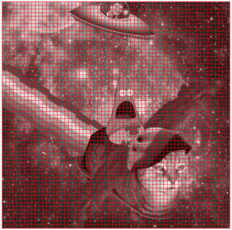


If the dimensions are not divisible in integer numbers of blocks, the image can be padded



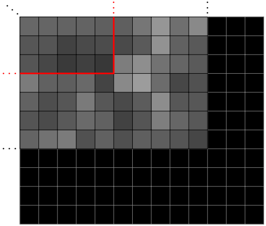
1[st] step: the input image is divided into non-overlapping 8×8 macro-blocks.

$\longrightarrow$



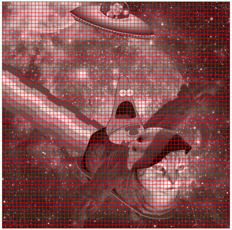
If the dimensions are not divisible in integer numbers of blocks, the image can be padded

→ with black pixels ■ (crude, may create noticeable artifacts).



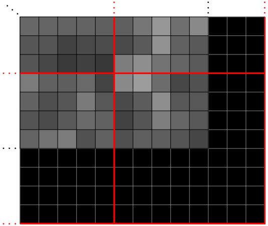
1^{st} step: the input image is divided into non-overlapping 8×8 macro-blocks.

$\longrightarrow$



If the dimensions are not divisible in integer numbers of blocks, the image can be padded

$\rightarrow$ with black pixels ■ (crude, may create noticeable artifacts).

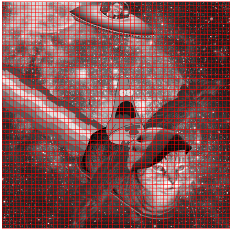


JPEG compression algorithm

Step 1: Block splitting

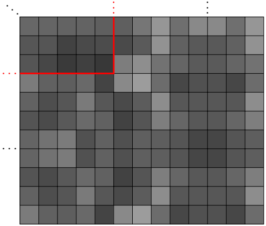
1^{st} step: the input image is divided into non-overlapping 8×8 macro-blocks.

$\longrightarrow$



If the dimensions are not divisible in integer numbers of blocks, the image can be padded

- $\rightarrow$ with black pixels ■ (crude, may create noticeable artifacts).
- $\rightarrow$ by replicating the border pixels (reduces the artifacts, but not necessarily all of them).

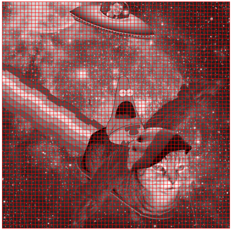


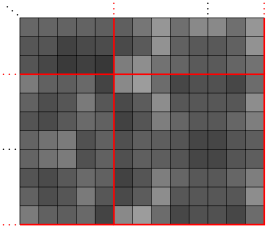
JPEG compression algorithm

Step 1: Block splitting

1st step: the input image is divided into non-overlapping 8×8 macro-blocks.

$\longrightarrow$



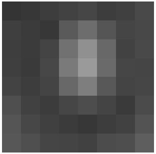


If the dimensions are not divisible in integer numbers of blocks, the image can be padded

- $\rightarrow$ with black pixels ■ (crude, may create noticeable artifacts).
- $\rightarrow$ by replicating the border pixels (reduces the artifacts, but not necessarily all of them).

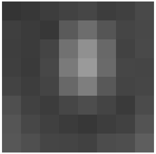
2[nd] step: the DCT of each 8×8 block is computed.

$$\mathbf{B}_i = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$



2^{nd} step: the DCT of each 8×8 block is computed.

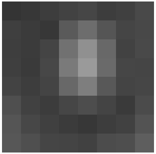
$$\mathbf{B}_i = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

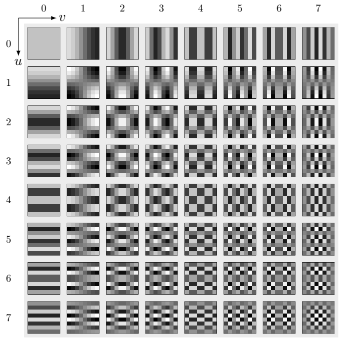


$$\mathcal{J}_{\mathbf{B}_i} = \mathrm{DCT}(\mathbf{B}_i) = \begin{bmatrix} 609 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

2^{nd} step: the DCT of each 8×8 block is computed.

$$\mathbf{B}_i = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$





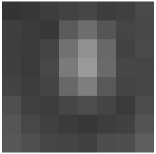
DC coefficient

$$\mathcal{J}_{\mathbf{B}_i} = \mathrm{DCT}(\mathbf{B}_i) = \begin{bmatrix} 609 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

AC coefficients

2^{nd} step: the DCT of each 8×8 block is computed.

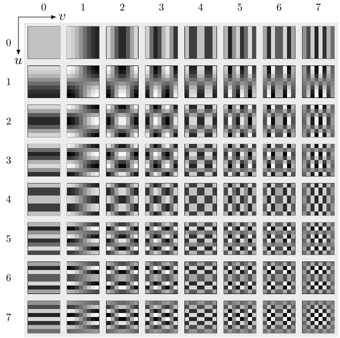
$$\mathbf{B}_i = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$



DC coefficient

$$\mathcal{J}_{\mathbf{B}_i} = \mathrm{DCT}(\mathbf{B}_i) = \begin{bmatrix} 609 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

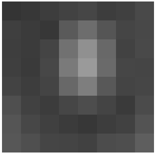
AC coefficients

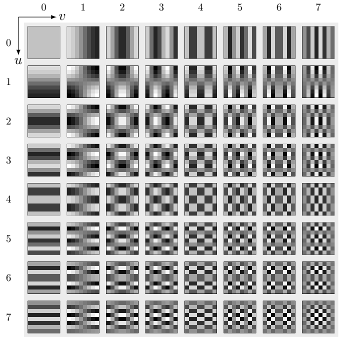


In average, $\mathbf{B}_i$ has a mean value close to 128 $\Rightarrow$ DC coefficient of $\mathrm{DCT}(\mathbf{B}_i - 128)$ should be close to 0.

2^{nd} step: the DCT of each (8×8 block) -128 is computed.

$$\mathbf{B}_i = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$





$$\mathcal{J}_{\mathbf{B}_i} = \mathrm{DCT}(\mathbf{B}_i - 128) = \begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

In average, $\mathbf{B}_i$ has a mean value close to 128 $\Rightarrow$ DC coefficient of $\mathrm{DCT}(\mathbf{B}_i - 128)$ should be close to 0.

3rd step: the DCT is quantized by some quantification matrix $\mathcal{Q}$.

$$\tilde{\mathcal{J}}_{\mathbf{B}_i} = \lfloor \mathcal{J}_{\mathbf{B}_i} \oslash \mathcal{Q} \rceil$$

element-wise division

3[rd] step: the DCT is quantized by some quantification matrix $\mathcal{Q}$.

$$\tilde{\mathcal{J}}_{\mathbf{B}_i} = \lfloor \mathcal{J}_{\mathbf{B}_i} \oslash \mathcal{Q} \rceil$$

element-wise division

$$= \left\lfloor \begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} \oslash \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \right\rceil$$

If you wonder how the hell did they come up with those quantization values, go read *JPEG: Still image data compression standard* by W.B. Pennebaker & J.L. Mitchell, Springer Science & Business Media (1992).

JPEG compression algorithm

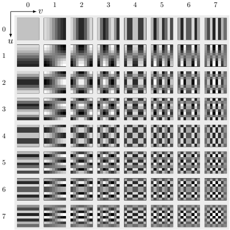
Step 3: quantization

3rd step: the DCT is quantized by some quantification matrix $\mathcal{Q}$.

$$\tilde{\mathcal{J}}_{\mathbf{B}_i} = \lfloor \mathcal{J}_{\mathbf{B}_i} \oslash \mathcal{Q} \rceil$$

element-wise division

$$= \left\lfloor \begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} \oslash \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \right\rceil$$



If you wonder how the hell did they come up with those quantization values, go read *JPEG: Still image data compression standard* by W.B. Pennebaker & J.L. Mitchell, Springer Science & Business Media (1992).

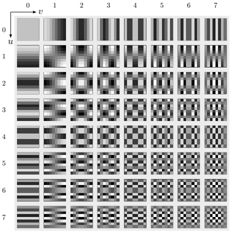
3[rd] step: the DCT is quantized by some quantification matrix $\mathcal{Q}$.

$$\tilde{\mathcal{J}}_{\mathbf{B}_i} = \lfloor \mathcal{J}_{\mathbf{B}_i} \oslash \mathcal{Q} \rceil$$

element-wise division

$$= \left\lfloor \begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} \oslash \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \right\rceil$$

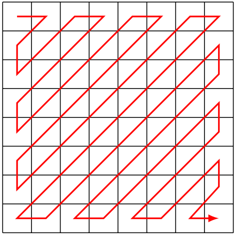
$$= \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



If you wonder how the hell did they come up with those quantization values, go read *JPEG: Still image data compression standard* by W.B. Pennebaker & J.L. Mitchell, Springer Science & Business Media (1992).

4th step: Arrange the quantized values in sequence following the zigzag order and use Huffman encoding with pre-determined conversion tables.

$$\tilde{\mathcal{J}}_{\mathbf{B}_i} = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

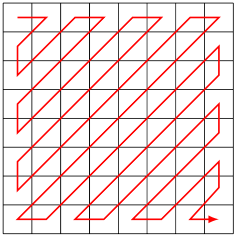

zigzag ordering

JPEG compression algorithm

Step 4: entropy coding

4^{th} step: Arrange the quantized values in sequence following the zigzag order and use Huffman encoding with pre-determined conversion tables.

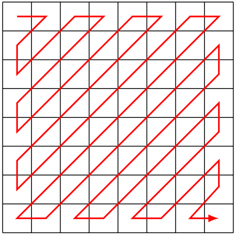
$$\mathrm{zigzag}(\tilde{\mathcal{J}}_{\mathbf{B}_i}) = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



zigzag ordering

4^{th} step: Arrange the quantized values in sequence following the zigzag order and use Huffman encoding with pre-determined conversion tables.

$$\text{zigzag}(\tilde{\mathcal{J}}_{\mathbf{B}_i}) = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

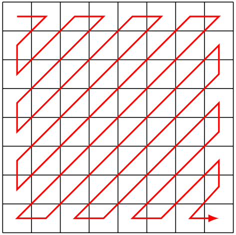


zigzag ordering

$=$ -26 -3 0 -3 -2 -6 2 -4 1 -3 1 1 5 1 2 -1 1 -1 2 0 0 0 0 0 -1 -1 EOB

4^{th} step: Arrange the quantized values in sequence following the zigzag order and use Huffman encoding with pre-determined conversion tables.

$$\text{zigzag}(\tilde{\mathcal{J}}_{\mathbf{B}_i}) = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



zigzag ordering

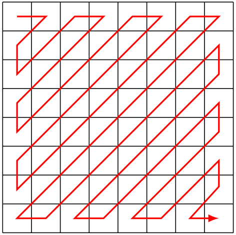
= -26 -3 0 -3 -2 -6 2 -4 1 -3 1 1 5 1 2 -1 1 -1 2 0 0 0 0 0 -1 -1 EOB

After applying Huffman with standard JPEG tables, the final encoding is:

1010110 0100 11100100 0101 100001 0110 100011 001 0100 001 001 100101 001 0110 000 001 000 0110 11110100 000 1010

4^{th} step: Arrange the quantized values in sequence following the zigzag order and use Huffman encoding with pre-determined conversion tables.

$$\text{zigzag}(\tilde{\mathcal{J}}_{\mathbf{B}_i}) = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



zigzag ordering

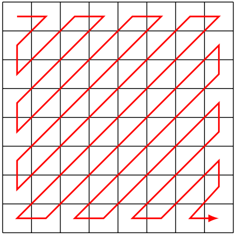
$= $ -26 -3 0 -3 -2 -6 2 -4 1 -3 1 1 5 1 2 -1 1 -1 2 0 0 0 0 0 -1 -1 EOB

After applying Huffman with standard JPEG tables, the final encoding is:

1010110 0100 11100100 0101 100001 0110 100011 001 0100 001 001 100101 001 0110 000 001 000 0110 11110100 000 1010

4^{th} step: Arrange the quantized values in sequence following the zigzag order and use Huffman encoding with pre-determined conversion tables.

$$\text{zigzag}(\tilde{\mathcal{J}}_{\mathbf{B}_i}) = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



zigzag ordering

= -26 -3 0 -3 -2 -6 2 -4 1 -3 1 1 5 1 2 -1 1 -1 2 0 0 0 0 0 -1 -1 EOB

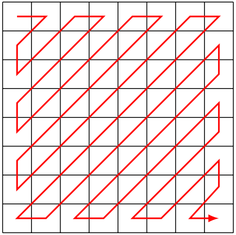
After applying Huffman with standard JPEG tables, the final encoding is:

1010110 0100 11100100 0101 100001 0110 100011 001 0100 001 001 100101 001 0110 000 001 000 0110 11110100 000 1010

Before compression: $8 \times 8 \times 8 = 512$ bits.
After compression: 94 bits.

4^{th} step: Arrange the quantized values in sequence following the zigzag order and use Huffman encoding with pre-determined conversion tables.

$$\text{zigzag}(\tilde{\mathcal{J}}_{\mathbf{B}_i}) = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



zigzag ordering

= -26 -3 0 -3 -2 -6 2 -4 1 -3 1 1 5 1 2 -1 1 -1 2 0 0 0 0 0 -1 -1 EOB

After applying Huffman with standard JPEG tables, the final encoding is:

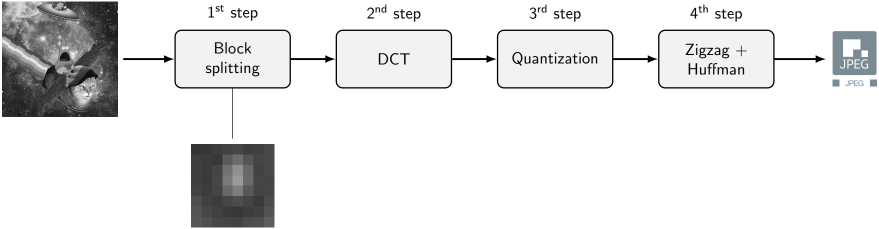
1010110 0100 11100100 0101 100001 0110 100011 001 0100 001 001 100101 001 0110 000 001 000 0110 11110100 000 1010

$$\left.\begin{array}{ll} \text{Before compression:} & 8 \times 8 \times 8 = 512 \text{ bits.} \\ \text{After compression:} & 94 \text{ bits.} \end{array}\right\} \Rightarrow \text{compression ratio} \cong 5.5$$

JPEG compression algorithm

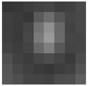
Summary of the compression scheme

→ 1st step: Block splitting → 2nd step: DCT → 3rd step: Quantization → 4th step: Zigzag + Huffman → JPEG



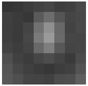
1st step
Block splitting
2nd step
DCT
3rd step
Quantization
4th step
Zigzag + Huffman
JPEG
JPEG

1^{st} step: Block splitting → 2^{nd} step: DCT → 3^{rd} step: Quantization → 4^{th} step: Zigzag + Huffman → JPEG



$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

1st step: Block splitting → 2nd step: DCT → 3rd step: Quantization → 4th step: Zigzag + Huffman → JPEG



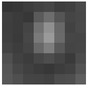
$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

JPEG compression algorithm

Summary of the compression scheme

1st step: Block splitting



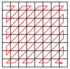
2nd step: DCT

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

3rd step: Quantization

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

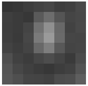
4th step: Zigzag + Huffman


zigzag ordering

-26 -3 0 -3 -2 -6 2 -4
1 -3 1 1 5 1 2 -1 1 -1
2 0 0 0 0 0 -1 -1 EOB

→ JPEG

1st step: **Block splitting**



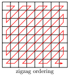
2nd step: **DCT**

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

3rd step: **Quantization**

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4th step: **Zigzag + Huffman**



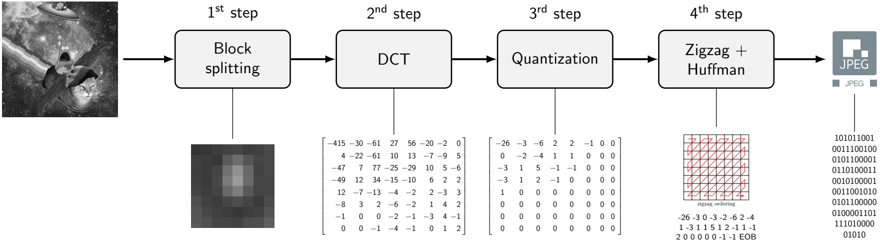
-26 -3 0 -3 -2 -6 2 -4
1 -3 1 1 5 1 2 -1 1 -1
2 0 0 0 0 0 -1 -1 EOB

JPEG

101011001
0011100100
0101100001
0110100011
0010100001
0011001010
0101100000
0100001101
111010000
01010

JPEG compression algorithm

Summary of the compression scheme




```
-rw-r--r-- 1 gtochon lrde 245K mai   19 13:22 randompic.tif
-rw-r--r-- 1 gtochon lrde  35K mai   19 14:18 randompic.jpg
```

JPEG decompression process is the exact inverse of the compression scheme.

→ Huffman^{-1} + Zigzag^{-1} → Reverse quantization → DCT^{-1} → Block combination →

JPEG compression algorithm

Decompression

JPEG decompression process is the exact inverse of the compression scheme.

JPEG → Huffman^{-1} + Zigzag^{-1} → Reverse quantization → DCT^{-1} → Block combination →

101011001001
110010001011
000010110100
011001010000
100110010100
101100000010
000110111101
000001010

JPEG compression algorithm

Decompression

JPEG decompression process is the exact inverse of the compression scheme.

JPEG → Huffman^{-1} + Zigzag^{-1} → Reverse quantization → DCT^{-1} → Block combination →

101011001001
110010001011
000010110100
011001010000
100110010100
101100000010
000110111101
000001010

→

-26 -3 0 -3 -2 -6 2 -4
1 -3 1 1 5 1 2 -1 1 -1
2 0 0 0 0 0 -1 -1 EOB

JPEG decompression process is the exact inverse of the compression scheme.

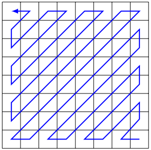
Huffman^{-1} + Zigzag^{-1} → Reverse quantization → DCT^{-1} → Block combination →

101011001001
110010001011
000010110100
011001010000
100110010100
101100000010
000110111101
000001010

→

-26 -3 0 -3 -2 -6 2 -4
1 -3 1 1 5 1 2 -1 1 -1
2 0 0 0 0 0 -1 -1 EOB

→



zigzag reverse ordering

JPEG decompression process is the exact inverse of the compression scheme.

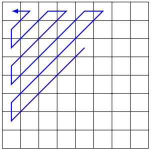
JPEG → Huffman^{-1} + Zigzag^{-1} → Reverse quantization → DCT^{-1} → Block combination →

101011001001
110010001011
000010110100
011001010000
100110010100
101100000010
000110111101
000001010

→

-26 -3 0 -3 -2 -6 2 -4
1 -3 1 1 5 1 2 -1 1 -1
2 0 0 0 0 0 -1 -1 EOB

→



zigzag reverse ordering

JPEG decompression process is the exact inverse of the compression scheme.

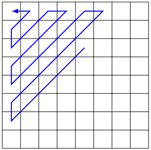
JPEG → **Huffman^{-1} + Zigzag^{-1}** → Reverse quantization → DCT^{-1} → Block combination →

101011001001
110010001011
000010110100
011001010000
100110010100
101100000010
000110111101
000001010

⟶

-26 -3 0 -3 -2 -6 2 -4
1 -3 1 1 5 1 2 -1 1 -1
2 0 0 0 0 0 -1 -1 EOB

⟶

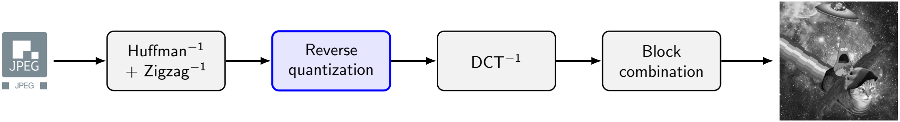


zigzag reverse ordering

⟶

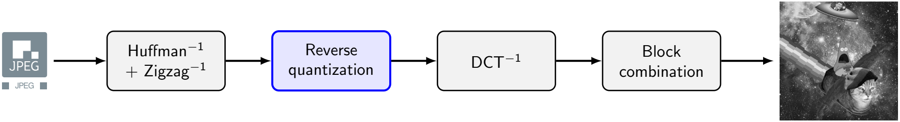
$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \tilde{\mathcal{J}}_{\mathbf{B}_i}$$

JPEG decompression process is the exact inverse of the compression scheme.



$\tilde{\mathcal{J}}_{\mathbf{B}_i^\star} = \tilde{\mathcal{J}}_{\mathbf{B}_i} \circ \mathcal{Q}$ Hadamard (component-wise) product

JPEG decompression process is the exact inverse of the compression scheme.



$$\tilde{\mathcal{J}}_{\mathbf{B}_i^\star} = \tilde{\mathcal{J}}_{\mathbf{B}_i} \circ \mathcal{Q}$$

Hadamard (component-wise) product

$$= \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \circ \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

JPEG compression algorithm

Decompression

JPEG decompression process is the exact inverse of the compression scheme.

JPEG → Huffman^{-1} + Zigzag^{-1} → **Reverse quantization** → DCT^{-1} → Block combination →

$\tilde{\mathcal{J}}_{\mathbf{B}_i^\star} = \tilde{\mathcal{J}}_{\mathbf{B}_i} \circ \mathcal{Q}$ Hadamard (component-wise) product

$$= \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \circ \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} = \begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -42 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

JPEG compression algorithm

Decompression

JPEG decompression process is the exact inverse of the compression scheme.

JPEG → Huffman^{-1} + Zigzag^{-1} → Reverse quantization → DCT^{-1} → Block combination →

$$\mathrm{DCT}^{-1}\left(\tilde{\mathcal{J}}_{\mathbf{B}_i^\star}\right) = \mathbf{B}_i^\star - 128 = \begin{bmatrix} -66 & -63 & -71 & -68 & -56 & -65 & -68 & -46 \\ -71 & -73 & -72 & -46 & -20 & -41 & -66 & -57 \\ -70 & -78 & -68 & -17 & 20 & -14 & -61 & -63 \\ -63 & -73 & -62 & -8 & 27 & -14 & -60 & -58 \\ -58 & -65 & -61 & -27 & -6 & -40 & -68 & -50 \\ -57 & -57 & -64 & -58 & -48 & -66 & -72 & -47 \\ -53 & -46 & -61 & -74 & -65 & -63 & -62 & -45 \\ -47 & -34 & -53 & -74 & -60 & -47 & -47 & -41 \end{bmatrix}$$

JPEG decompression process is the exact inverse of the compression scheme.

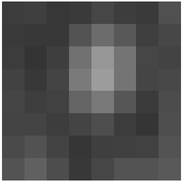
JPEG → Huffman^{-1} + Zigzag^{-1} → Reverse quantization → DCT^{-1} → Block combination →

$$\mathbf{B}_i^\star = \mathrm{DCT}^{-1}\left(\tilde{\mathcal{J}}_{\mathbf{B}_i^\star}\right) + 128 = \begin{bmatrix} 62 & 65 & 57 & 60 & 72 & 63 & 60 & 82 \\ 57 & 55 & 56 & 82 & 108 & 87 & 62 & 71 \\ 58 & 50 & 60 & 111 & 148 & 114 & 67 & 65 \\ 65 & 55 & 66 & 120 & 155 & 114 & 68 & 70 \\ 70 & 63 & 67 & 101 & 122 & 88 & 60 & 78 \\ 71 & 71 & 64 & 70 & 80 & 62 & 56 & 81 \\ 75 & 82 & 67 & 54 & 63 & 65 & 66 & 83 \\ 81 & 94 & 75 & 54 & 68 & 81 & 81 & 87 \end{bmatrix}$$

JPEG decompression process is the exact inverse of the compression scheme.

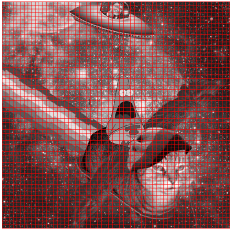
JPEG → Huffman^{-1} + Zigzag^{-1} → Reverse quantization → DCT^{-1} → Block combination →

$$\mathbf{B}_i^\star = \mathrm{DCT}^{-1}\left(\tilde{\mathcal{J}}_{\mathbf{B}_i^\star}\right) + 128 = \begin{bmatrix} 62 & 65 & 57 & 60 & 72 & 63 & 60 & 82 \\ 57 & 55 & 56 & 82 & 108 & 87 & 62 & 71 \\ 58 & 50 & 60 & 111 & 148 & 114 & 67 & 65 \\ 65 & 55 & 66 & 120 & 155 & 114 & 68 & 70 \\ 70 & 63 & 67 & 101 & 122 & 88 & 60 & 78 \\ 71 & 71 & 64 & 70 & 80 & 62 & 56 & 81 \\ 75 & 82 & 67 & 54 & 63 & 65 & 66 & 83 \\ 81 & 94 & 75 & 54 & 68 & 81 & 81 & 87 \end{bmatrix}$$



JPEG decompression process is the exact inverse of the compression scheme.

JPEG → Huffman^{-1} + Zigzag^{-1} → Reverse quantization → DCT^{-1} → **Block combination** →



$\longrightarrow$

Lossy compression indeed...

... but what is lost in the process?

The only lossy operation in the JPEG process is the rounding $\lfloor \cdot \rceil$ during quantization.

$$\tilde{\mathcal{J}}_{\mathbf{B}_i^\star} = \lfloor \mathcal{J}_{\mathbf{B}_i} \oslash \mathcal{Q} \rceil \circ \mathcal{Q} = \begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -42 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \neq \begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} = \mathcal{J}_{\mathbf{B}_i}$$

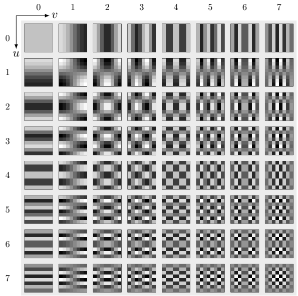
Lossy compression indeed...

... but what is lost in the process?

The only lossy operation in the JPEG process is the rounding $\lfloor \cdot \rceil$ during quantization.

$$\tilde{\mathcal{J}}_{\mathbf{B}_i^\star} = \lfloor \mathcal{J}_{\mathbf{B}_i} \oslash \mathcal{Q} \rceil \circ \mathcal{Q} = \begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -42 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \neq \begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} = \mathcal{J}_{\mathbf{B}_i}$$

$$\Rightarrow \mathcal{J}_{\mathbf{B}_i} - \tilde{\mathcal{J}}_{\mathbf{B}_i^\star} = \begin{bmatrix} 1 & 3 & -1 & -5 & 8 & 20 & -2 & 0 \\ 4 & 2 & -5 & -9 & -13 & -7 & -9 & 5 \\ -5 & -6 & -3 & -1 & 11 & 10 & 5 & -6 \\ 7 & -5 & -10 & 14 & -10 & 6 & 2 & 2 \\ -6 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

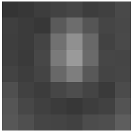


Lossy compression indeed...

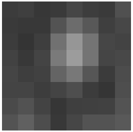
... but what is lost in the process?

The only lossy operation in the JPEG process is the rounding $\lfloor \cdot \rceil$ during quantization.

$$\mathbf{B}_i = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

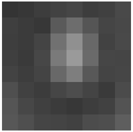


$$\mathbf{B}_i^\star = \begin{bmatrix} 60 & 63 & 55 & 58 & 70 & 61 & 58 & 80 \\ 58 & 56 & 56 & 83 & 108 & 88 & 63 & 71 \\ 60 & 52 & 62 & 113 & 150 & 116 & 70 & 67 \\ 66 & 56 & 68 & 122 & 156 & 116 & 69 & 72 \\ 69 & 62 & 65 & 100 & 120 & 86 & 59 & 76 \\ 68 & 68 & 61 & 68 & 78 & 60 & 53 & 78 \\ 74 & 82 & 67 & 54 & 63 & 64 & 65 & 83 \\ 83 & 96 & 77 & 56 & 70 & 83 & 83 & 89 \end{bmatrix}$$

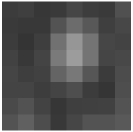


The only lossy operation in the JPEG process is the rounding $\lfloor \cdot \rceil$ during quantization.

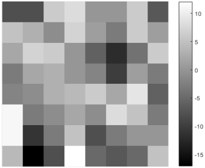
$$\mathbf{B}_i = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$



$$\mathbf{B}_i^\star = \begin{bmatrix} 60 & 63 & 55 & 58 & 70 & 61 & 58 & 80 \\ 58 & 56 & 56 & 83 & 108 & 88 & 63 & 71 \\ 60 & 52 & 62 & 113 & 150 & 116 & 70 & 67 \\ 66 & 56 & 68 & 122 & 156 & 116 & 69 & 72 \\ 69 & 62 & 65 & 100 & 120 & 86 & 59 & 76 \\ 68 & 68 & 61 & 68 & 78 & 60 & 53 & 78 \\ 74 & 82 & 67 & 54 & 63 & 64 & 65 & 83 \\ 83 & 96 & 77 & 56 & 70 & 83 & 83 & 89 \end{bmatrix}$$



$$\Rightarrow \epsilon_i = \mathbf{B}_i - \mathbf{B}_i^\star = \begin{bmatrix} -8 & -8 & 6 & 8 & 0 & 0 & 6 & -7 \\ 5 & 3 & -1 & 7 & 1 & -3 & 6 & 1 \\ 2 & 7 & 6 & 0 & -6 & -12 & -4 & 6 \\ -3 & 2 & 3 & 0 & -2 & -10 & 1 & -3 \\ -2 & -1 & 3 & 4 & 6 & 2 & 9 & -6 \\ 11 & -3 & -1 & 2 & -1 & 8 & 5 & -3 \\ 11 & -11 & -3 & 5 & -8 & -3 & 0 & 0 \\ 4 & -17 & -8 & 12 & -5 & -7 & -5 & 5 \end{bmatrix}$$

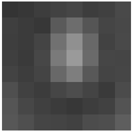


Lossy compression indeed...

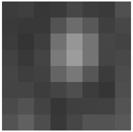
... but what is lost in the process?

The only lossy operation in the JPEG process is the rounding $\lfloor \cdot \rceil$ during quantization.

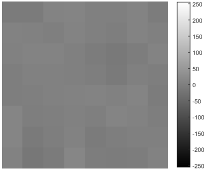
$$\mathbf{B}_i = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$



$$\mathbf{B}_i^{\star} = \begin{bmatrix} 60 & 63 & 55 & 58 & 70 & 61 & 58 & 80 \\ 58 & 56 & 56 & 83 & 108 & 88 & 63 & 71 \\ 60 & 52 & 62 & 113 & 150 & 116 & 70 & 67 \\ 66 & 56 & 68 & 122 & 156 & 116 & 69 & 72 \\ 69 & 62 & 65 & 100 & 120 & 86 & 59 & 76 \\ 68 & 68 & 61 & 68 & 78 & 60 & 53 & 78 \\ 74 & 82 & 67 & 54 & 63 & 64 & 65 & 83 \\ 83 & 96 & 77 & 56 & 70 & 83 & 83 & 89 \end{bmatrix}$$



$$\Rightarrow \epsilon_i = \mathbf{B}_i - \mathbf{B}_i^{\star} = \begin{bmatrix} -8 & -8 & 6 & 8 & 0 & 0 & 6 & -7 \\ 5 & 3 & -1 & 7 & 1 & -3 & 6 & 1 \\ 2 & 7 & 6 & 0 & -6 & -12 & -4 & 6 \\ -3 & 2 & 3 & 0 & -2 & -10 & 1 & -3 \\ -2 & -1 & 3 & 4 & 6 & 2 & 9 & -6 \\ 11 & -3 & -1 & 2 & -1 & 8 & 5 & -3 \\ 11 & -11 & -3 & 5 & -8 & -3 & 0 & 0 \\ 4 & -17 & -8 & 12 & -5 & -7 & -5 & 5 \end{bmatrix}$$

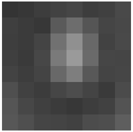


Lossy compression indeed...

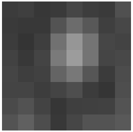
... but what is lost in the process?

The only lossy operation in the JPEG process is the rounding $\lfloor\cdot\rceil$ during quantization.

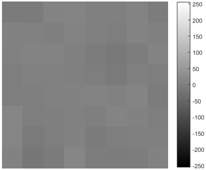
$$\mathbf{B}_i = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$



$$\mathbf{B}_i^\star = \begin{bmatrix} 60 & 63 & 55 & 58 & 70 & 61 & 58 & 80 \\ 58 & 56 & 56 & 83 & 108 & 88 & 63 & 71 \\ 60 & 52 & 62 & 113 & 150 & 116 & 70 & 67 \\ 66 & 56 & 68 & 122 & 156 & 116 & 69 & 72 \\ 69 & 62 & 65 & 100 & 120 & 86 & 59 & 76 \\ 68 & 68 & 61 & 68 & 78 & 60 & 53 & 78 \\ 74 & 82 & 67 & 54 & 63 & 64 & 65 & 83 \\ 83 & 96 & 77 & 56 & 70 & 83 & 83 & 89 \end{bmatrix}$$



$$\Rightarrow \epsilon_i = \mathbf{B}_i - \mathbf{B}_i^\star = \begin{bmatrix} -8 & -8 & 6 & 8 & 0 & 0 & 6 & -7 \\ 5 & 3 & -1 & 7 & 1 & -3 & 6 & 1 \\ 2 & 7 & 6 & 0 & -6 & -12 & -4 & 6 \\ -3 & 2 & 3 & 0 & -2 & -10 & 1 & -3 \\ -2 & -1 & 3 & 4 & 6 & 2 & 9 & -6 \\ 11 & -3 & -1 & 2 & -1 & 8 & 5 & -3 \\ 11 & -11 & -3 & 5 & -8 & -3 & 0 & 0 \\ 4 & -17 & -8 & 12 & -5 & -7 & -5 & 5 \end{bmatrix}$$



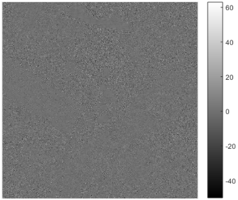
RMSE $= 6.01$
SNR $= 22.37$ dB

Reconstruction error at the image scale

The notorious compression artifacts of JPEG

$\mathcal{I}_{\text{TIF}}$

$\mathcal{I}_{\text{JPEG}}$

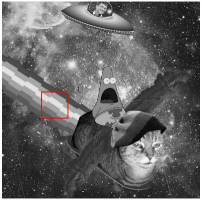


$\epsilon_{\text{JPEG}} = \mathcal{I}_{\text{TIF}} - \mathcal{I}_{\text{JPEG}}$

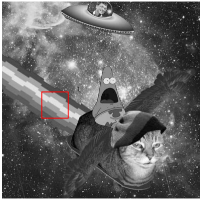
RMSE $= 6.71$
SNR $= 24.29$ dB

Reconstruction error at the image scale

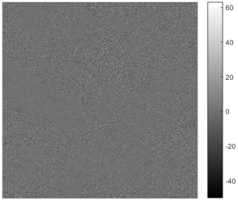
The notorious compression artifacts of JPEG



$\mathcal{I}_{\text{TIF}}$

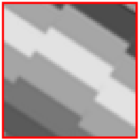


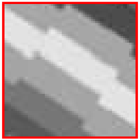
$\mathcal{I}_{\text{JPEG}}$



$\epsilon_{\text{JPEG}} = \mathcal{I}_{\text{TIF}} - \mathcal{I}_{\text{JPEG}}$

RMSE $= 6.71$
SNR $= 24.29$ dB

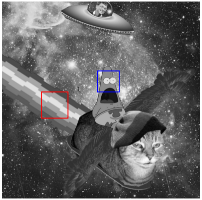




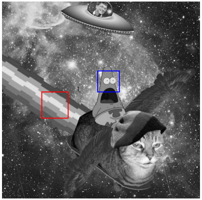
“mosaic effect” in homogeneous areas.

Reconstruction error at the image scale

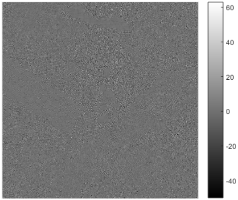
The notorious compression artifacts of JPEG



$\mathcal{I}_{\text{TIF}}$

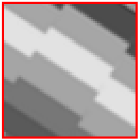


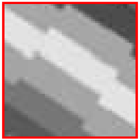
$\mathcal{I}_{\text{JPEG}}$



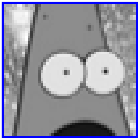
$\epsilon_{\text{JPEG}} = \mathcal{I}_{\text{TIF}} - \mathcal{I}_{\text{JPEG}}$

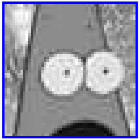
RMSE $= 6.71$
SNR $= 24.29$ dB





"mosaic effect" in homogeneous areas.





"ringing effect" on sharp edges.

Changing the compression quality

It is possible to adjust the compression quality of JPEG by modyfing the quantization matrix

Input: $\mathcal{Q}$, $q \in [1:100]$
Output: $\mathcal{Q}_q$
if $q < 50$ **then**
  $\alpha = \frac{5000}{q}$;
else
  $\alpha = 200 - 2q$;
end
$\mathcal{Q}_q = \left\lfloor \frac{\alpha\mathcal{Q} + 50}{100} \right\rfloor$;

$$\mathcal{Q} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Changing the compression quality

It is possible to adjust the compression quality of JPEG by modyfing the quantization matrix

Input: $\mathcal{Q}$, $q \in [1:100]$
Output: $\mathcal{Q}_q$
if $q < 50$ **then**
$\quad \alpha = \dfrac{5000}{q}$;
else
$\quad \alpha = 200 - 2q$;
end
$\mathcal{Q}_q = \left\lfloor \dfrac{\alpha\mathcal{Q} + 50}{100} \right\rfloor$;

$$\mathcal{Q} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$\mathcal{Q}_{25} = \begin{bmatrix} 32 & 22 & 20 & 32 & 48 & 80 & 102 & 122 \\ 24 & 24 & 28 & 38 & 52 & 116 & 120 & 110 \\ 28 & 26 & 32 & 48 & 80 & 114 & 138 & 112 \\ 28 & 34 & 44 & 58 & 102 & 174 & 160 & 124 \\ 36 & 44 & 74 & 112 & 136 & 218 & 206 & 154 \\ 48 & 70 & 110 & 128 & 162 & 208 & 226 & 184 \\ 98 & 128 & 156 & 174 & 206 & 242 & 240 & 202 \\ 144 & 184 & 190 & 196 & 224 & 200 & 206 & 198 \end{bmatrix} \qquad \mathcal{Q}_{50} = \mathcal{Q} \qquad \mathcal{Q}_{75} = \begin{bmatrix} 8 & 6 & 5 & 8 & 12 & 20 & 26 & 31 \\ 6 & 6 & 7 & 10 & 13 & 29 & 30 & 28 \\ 7 & 7 & 8 & 12 & 20 & 29 & 35 & 28 \\ 7 & 9 & 11 & 15 & 26 & 44 & 40 & 31 \\ 9 & 11 & 19 & 28 & 34 & 55 & 52 & 39 \\ 12 & 18 & 28 & 32 & 41 & 52 & 57 & 46 \\ 25 & 32 & 39 & 44 & 52 & 61 & 60 & 51 \\ 36 & 46 & 48 & 49 & 56 & 50 & 52 & 50 \end{bmatrix}$$

Influence of the quality factor (1/2)

$q = 50$

$$\begin{aligned} \text{RMSE} &= 6.71 \\ \text{SNR} &= 24.29 \text{ dB} \end{aligned}$$

```
-rw-r--r--  35K randompic_Q50.jpg
```

$q = 25$

$q = 50$

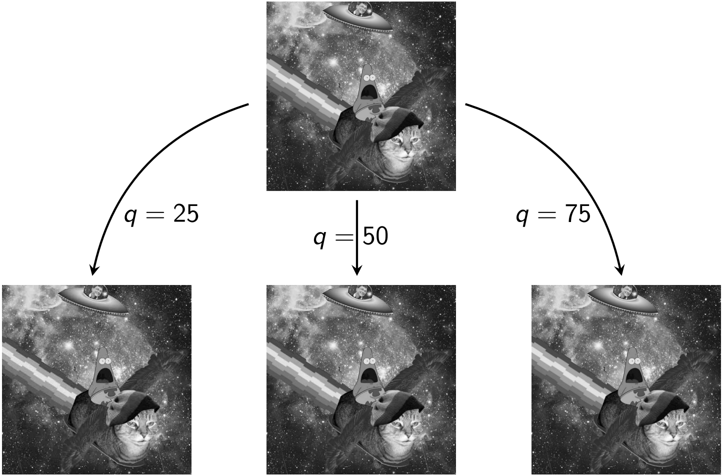
RMSE $= 8.83$
SNR $= 21.90$ dB

`-rw-r--r-- 21K randompic_Q25.jpg`

RMSE $= 6.71$
SNR $= 24.29$ dB

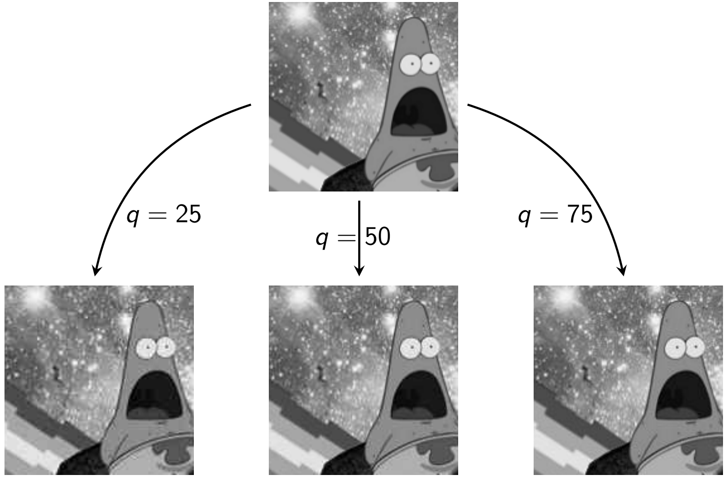
`-rw-r--r-- 35K randompic_Q50.jpg`

Influence of the quality factor (1/2)



$q = 25$	$q = 50$	$q = 75$
RMSE $= 8.83$	RMSE $= 6.71$	RMSE $= 4.82$
SNR $= 21.90$ dB	SNR $= 24.29$ dB	SNR $= 27.16$ dB
`-rw-r--r-- 21K randompic_Q25.jpg`	`-rw-r--r-- 35K randompic_Q50.jpg`	`-rw-r--r-- 52K randompic_Q75.jpg`

Influence of the quality factor (1/2)



RMSE $= 8.83$
SNR $= 21.90$ dB
`-rw-r--r--  21K randompic_Q25.jpg`

RMSE $= 6.71$
SNR $= 24.29$ dB
`-rw-r--r--  35K randompic_Q50.jpg`

RMSE $= 4.82$
SNR $= 27.16$ dB
`-rw-r--r--  52K randompic_Q75.jpg`

Influence of the quality factor (2/2)