



Xmas Research

Maria Shevchuk



Applying Gaussian Low Pass & High Pass filters

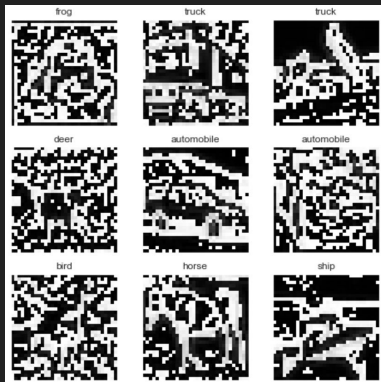
```
from scipy.ndimage import gaussian_filter
```

```
X_train = np.array([gaussian_filter(image, sigma = 1) for image in X_train])  
X_test = np.array([gaussian_filter(image, sigma = 1) for image in X_test])
```

```
X_train = np.array([image - gaussian_filter(image, sigma = 1) for image in X_train])  
X_test = np.array([image - gaussian_filter(image, sigma = 1) for image in X_test])
```

- Takes average over the nearby pixels (sigma = # of contributing pixels)

High Pass

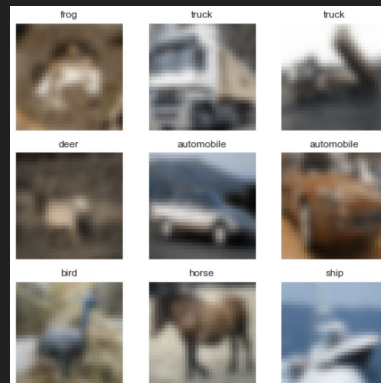
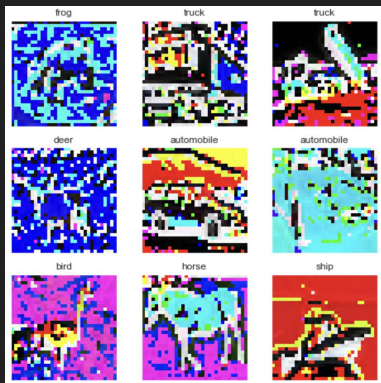


Grey

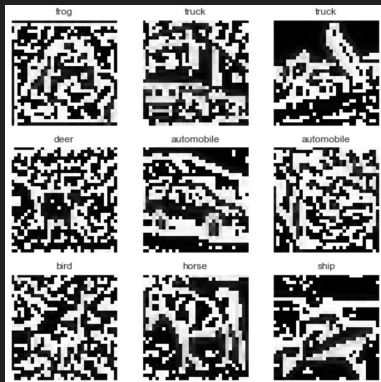
Low Pass



RGB



High Pass



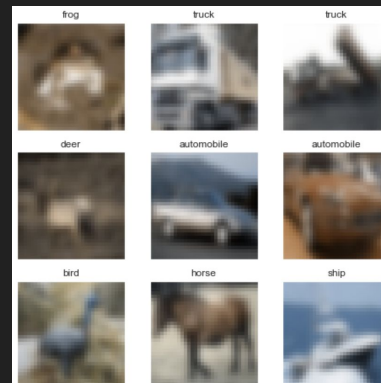
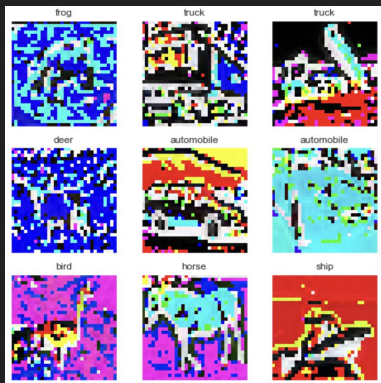
Low Pass



Grey

RGB

?



Averaging | cv.blur

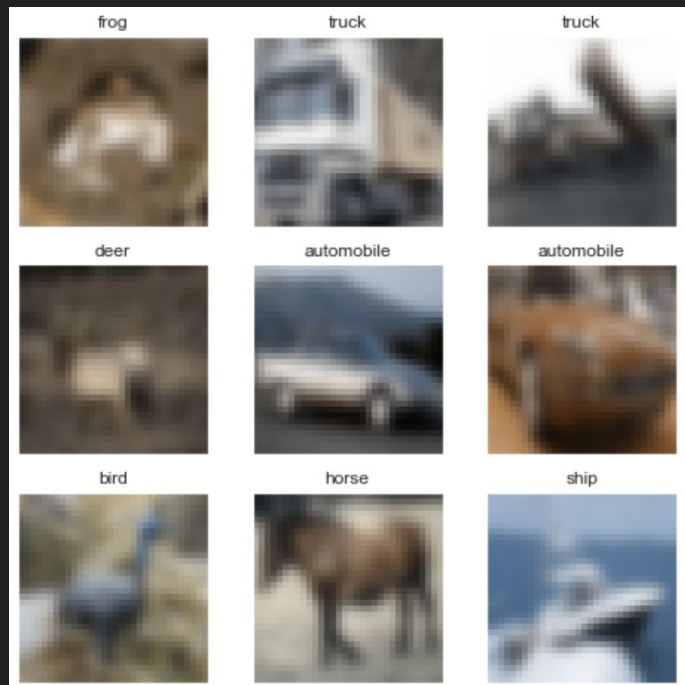
```
X_train = np.array([cv2.blur(image, (3,3)) for image in X_train])  
X_test = np.array([cv2.blur(image, (3,3)) for image in X_test])
```

```
X_train = np.array([image - cv2.blur(image, (3,3)) for image in X_train])  
X_test = np.array([image - cv2.blur(image, (3,3)) for image in X_test])
```

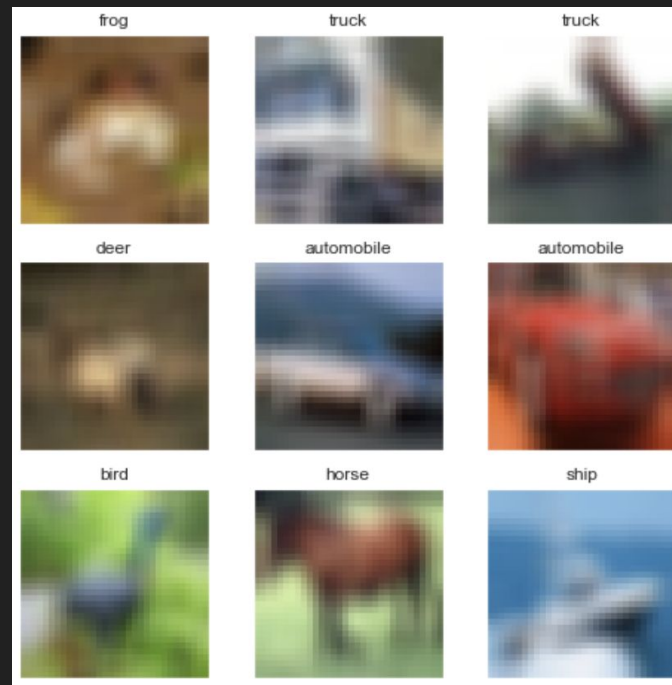
- Convolutes an image with a normalized box filter of specified size

$$K = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

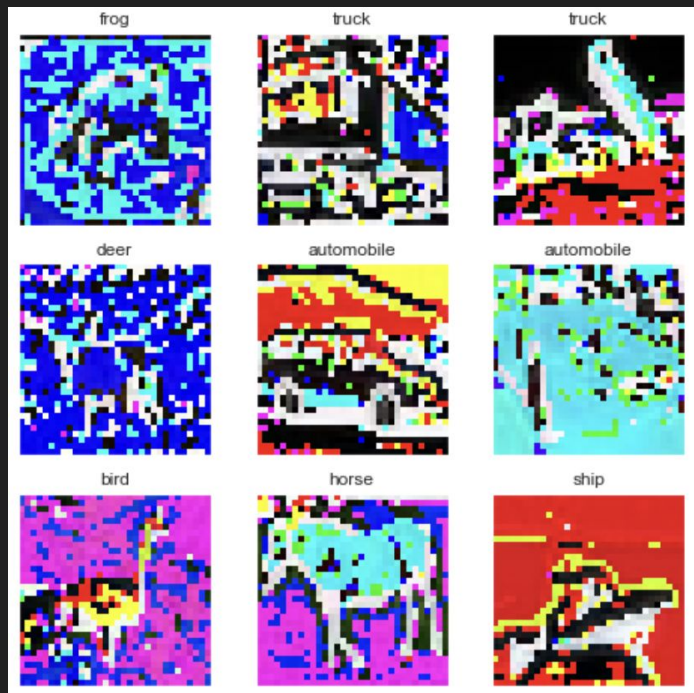
gaussian_filter



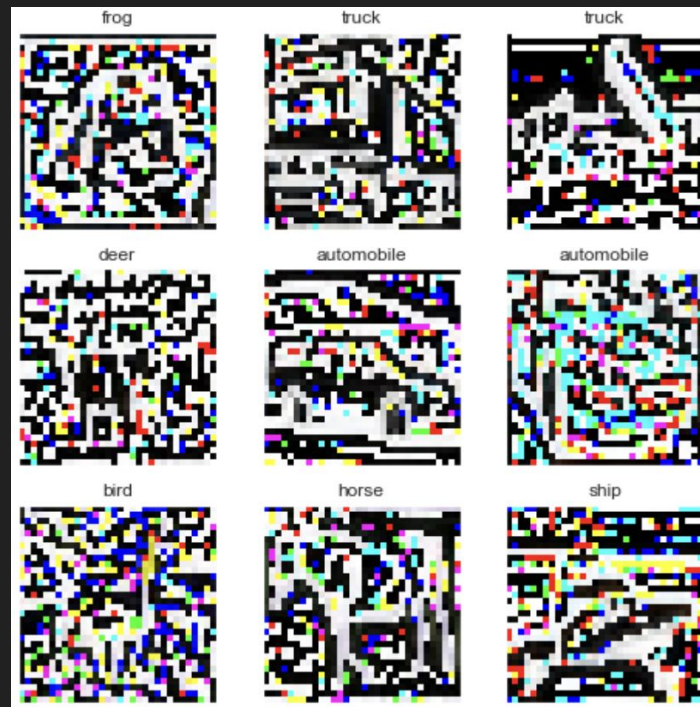
cv2.blur



gaussian_filter

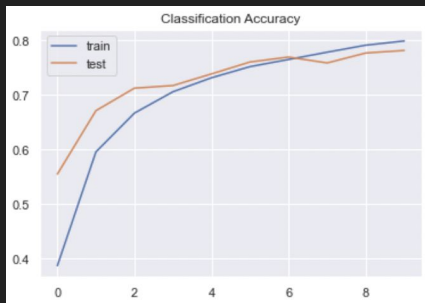


cv2.blur



Original

Accuracy: 0.7815
Loss: 0.6574



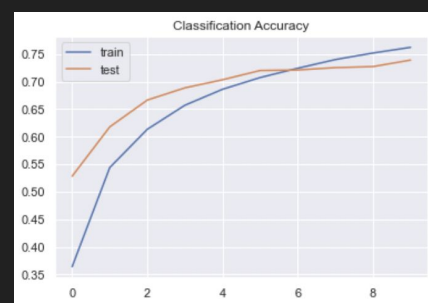
High Pass

Accuracy: 0.6684
Loss: 0.9668



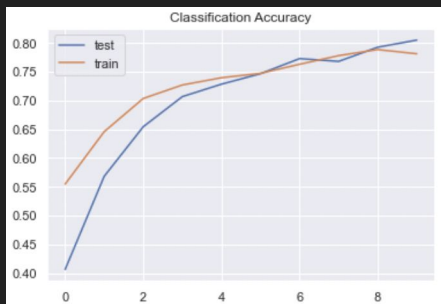
Low Pass

Accuracy: 0.7389
Loss: 0.7620

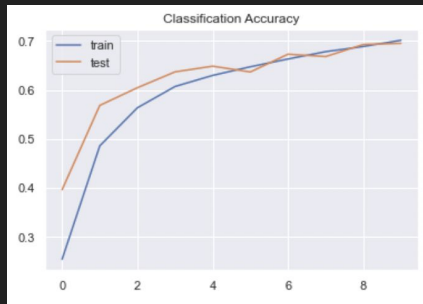


Grey

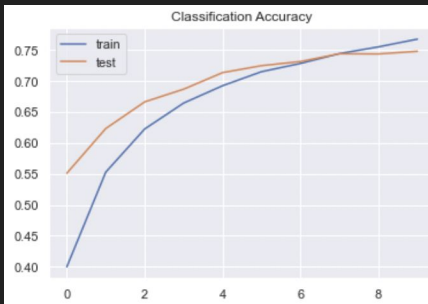
Accuracy: 0.7782
Loss: 0.6643



Accuracy: 0.6949
Loss: 0.9119



Accuracy: 0.7482
Loss: 0.7404



RGB

GREYSCALE

Original

High Pass

Low Pass

Greyscale Network Confusion Matrix

Actual \ Predicted	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	668	3	48	42	44	13	22	11	105	44
automobile	2	811	4	19	5	11	22	3	32	91
bird	37	1	613	55	109	88	65	24	7	1
cat	5	0	39	592	47	187	85	32	4	9
deer	2	1	35	70	743	36	72	33	8	0
dog	0	0	25	109	34	777	20	29	5	1
frog	1	0	22	41	11	38	875	2	8	2
horse	3	1	22	42	31	82	7	805	3	4
ship	21	5	13	15	4	8	14	5	891	24
truck	6	30	3	20	2	7	11	8	17	896

High Pass Greyscale Network Confusion Matrix

Actual \ Predicted	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	682	16	55	31	44	7	15	27	88	35
automobile	13	852	4	4	6	6	3	1	17	94
bird	78	6	418	71	213	93	48	40	17	16
cat	22	20	41	461	120	155	56	58	19	48
deer	33	6	35	43	681	31	32	99	9	31
dog	8	8	30	145	90	591	17	89	5	17
frog	22	28	21	65	156	26	631	16	16	19
horse	9	3	18	33	71	60	5	766	7	28
ship	92	49	13	13	14	5	9	9	726	70
truck	12	63	3	10	9	1	4	12	28	858

Low Pass Greyscale Network Confusion Matrix

Actual \ Predicted	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	762	12	57	14	35	6	10	8	65	31
automobile	14	846	4	4	6	1	16	0	17	92
bird	55	5	656	44	112	37	61	13	16	1
cat	26	7	89	532	92	102	83	29	15	25
deer	10	3	91	35	745	19	60	29	7	1
dog	8	5	75	185	67	550	46	41	16	7
frog	6	7	49	50	31	7	828	0	13	9
horse	16	3	48	39	87	36	10	741	7	13
ship	30	17	13	16	14	2	9	3	880	16
truck	27	61	5	17	8	4	11	9	21	837

RGB

Original

RGB Network Confusion Matrix

Actual \ Predicted	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	801	10	25	25	35	5	9	15	29	46
automobile	7	888	2	6	3	2	5	0	12	75
bird	56	0	536	60	120	84	106	23	9	6
cat	10	2	16	608	74	204	57	17	7	5
deer	4	1	17	40	833	24	44	34	3	0
dog	6	2	10	110	53	777	17	20	1	4
frog	4	1	13	29	35	21	893	2	1	1
horse	6	1	8	45	79	58	2	790	1	10
ship	65	15	2	20	9	3	5	5	854	22
truck	11	40	2	16	3	3	8	9	13	895

High Pass

High Pass Color Network Confusion Matrix

Actual \ Predicted	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	669	11	65	14	47	5	23	16	125	25
automobile	16	801	3	9	5	5	16	4	48	93
bird	61	4	473	57	192	67	85	29	25	7
cat	19	4	58	459	134	135	105	45	26	15
deer	15	3	44	29	742	29	50	59	16	13
dog	10	2	58	164	93	534	31	85	15	8
frog	14	6	26	61	100	14	753	4	18	4
horse	8	0	23	31	107	52	14	750	9	6
ship	65	19	18	11	10	3	12	8	829	25
truck	18	42	10	12	10	3	6	14	47	838

Low Pass

Low Pass Color Network Confusion Matrix

Actual \ Predicted	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	808	18	28	14	13	1	4	6	69	39
automobile	6	915	4	9	1	2	4	0	12	47
bird	77	6	641	45	101	42	50	21	11	6
cat	28	10	81	571	60	132	58	28	16	16
deer	32	3	75	58	722	25	35	41	5	4
dog	21	3	62	180	47	608	19	46	5	9
frog	12	9	53	53	44	16	794	5	7	7
horse	19	4	42	46	56	45	4	770	3	11
ship	54	27	11	8	5	5	4	2	863	21
truck	29	104	9	17	6	6	3	10	16	800

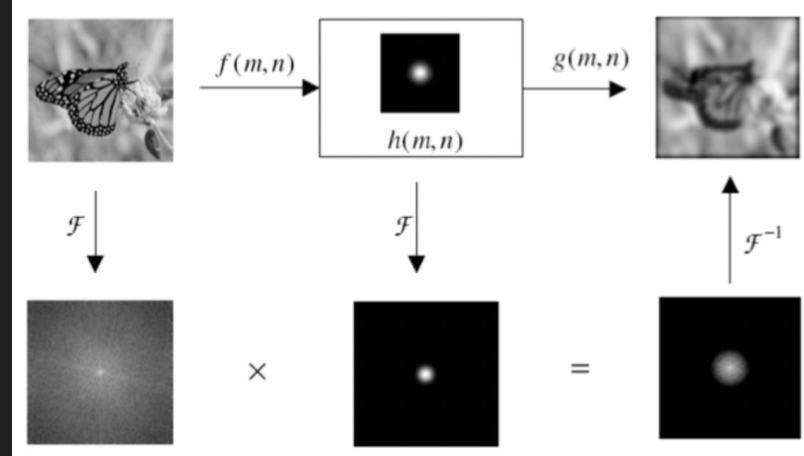
FFT

Image Processing in the Frequency Domain

The **Fourier Transform** is used to decompose an image into its sine and cosine components.

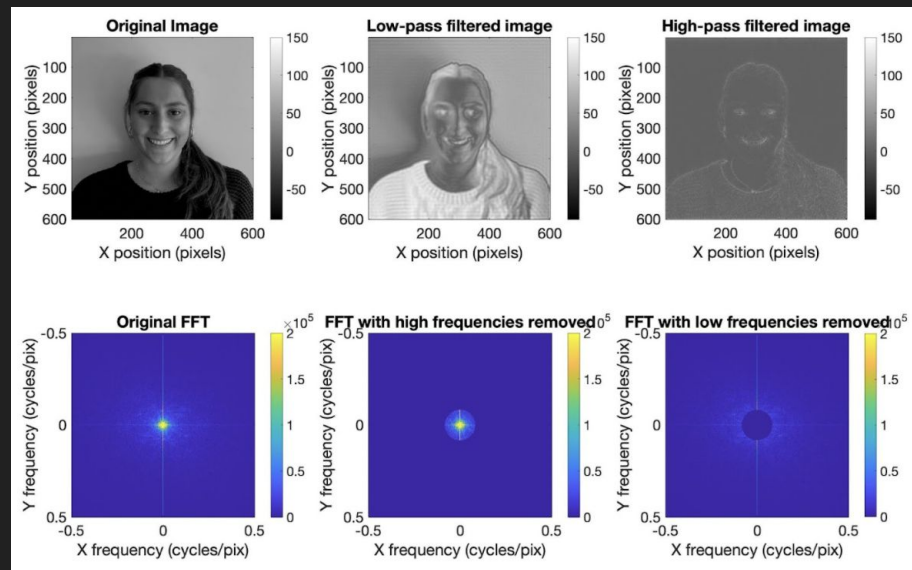
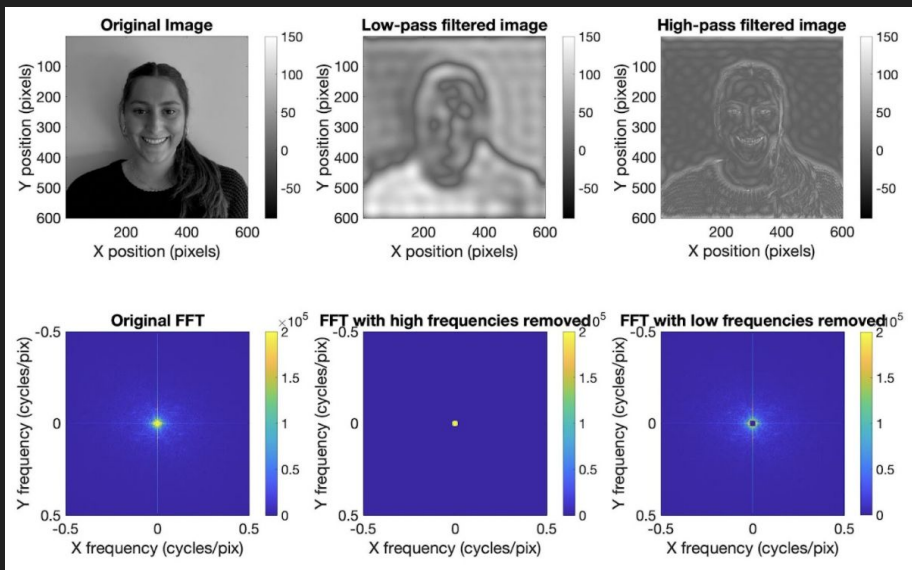
Theory

- A Fast Fourier Transform (FFT) is used for calculation of 2D Discrete Fourier Transform (DFT) which is used to find the frequency domain.
- When working with image data, the edge points represent the high frequencies, since the amplitude of a signal varies drastically at that point.



Mask size

Changing mask size allows for regulation of the filtered frequencies, basically controls the range of frequencies that are removed. All values either inside or outside the desired radius are zeroed.



Why is it useful?

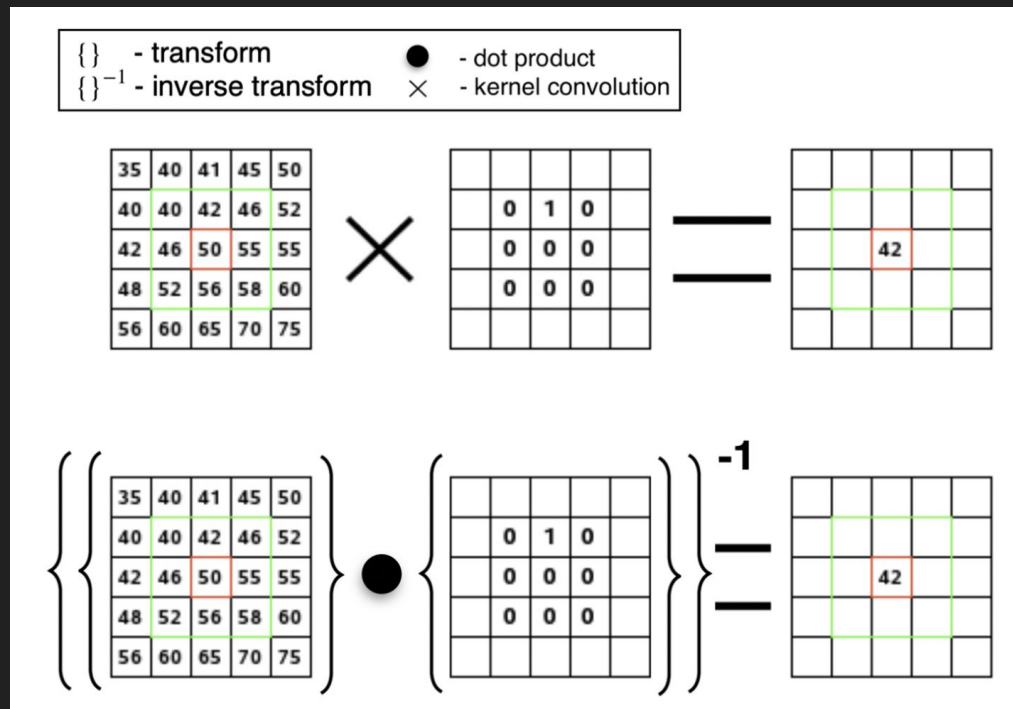
- Frequency domain gives you control over the whole images, where you can enhance (eg edges) and suppress (eg smooth shadow) different characteristics of the image very easily.
- Frequency domain has a established suit of processes and tools that be borrowed directly from signal processing in other domains.
- Some tools used for even image recognition such as correlation, convolution etc are much simpler and computationally cheaper in frequency domain.

More FT fun facts

The dot product of x and y within the Fourier domain is the convolution of the Fourier domains of x and y .

In 2D, the Fourier operation costs $O(N^2 \log N^2) = O(N^2 \log N)$. Usually, the filter $w \in \mathbb{R}^{K \times K}$ is a *sliding window* over the image, for a total of N^2 (K, K) convolutions that cost $O(N^2 K^2)$. However, by padding w to be size $N \times N$ and computing, we get the following:

Researchers have reported significant reductions in training time (**55s \rightarrow 0.4s!!!**) with an improved $O(N^2 K^2) \rightarrow O(N^2 \log N)$ convolution step.



$$x * y = F_N(F_N^{-1}(x) \bullet F_N^{-1}(y))$$