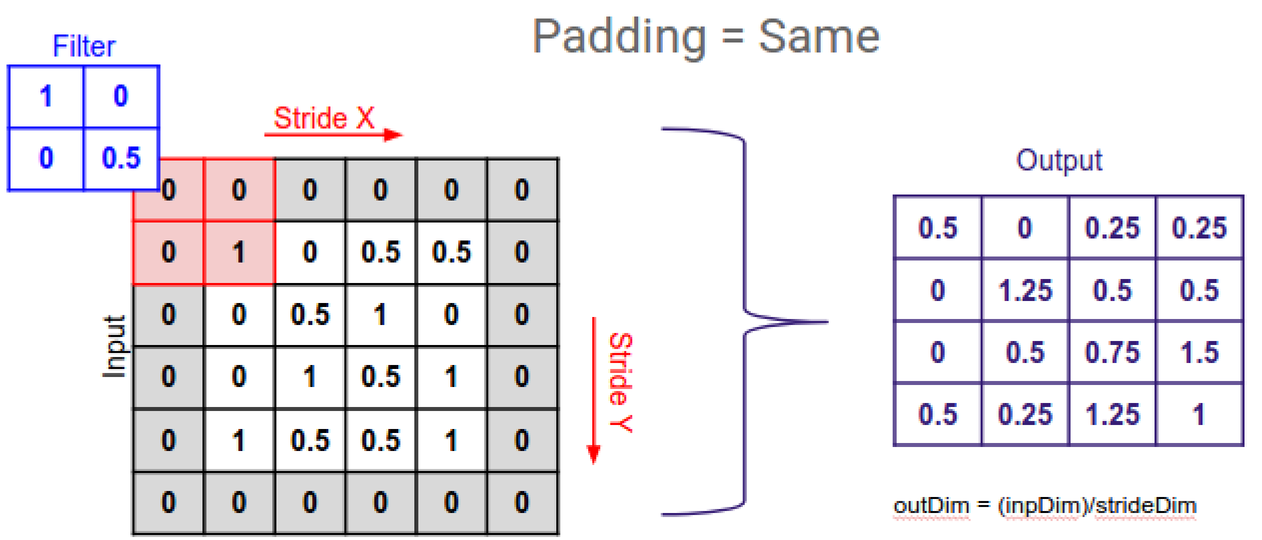
AlexNet

Computer Vision & Augmented Reality 연구실
학부연구생 강 준 구

2021-11-24

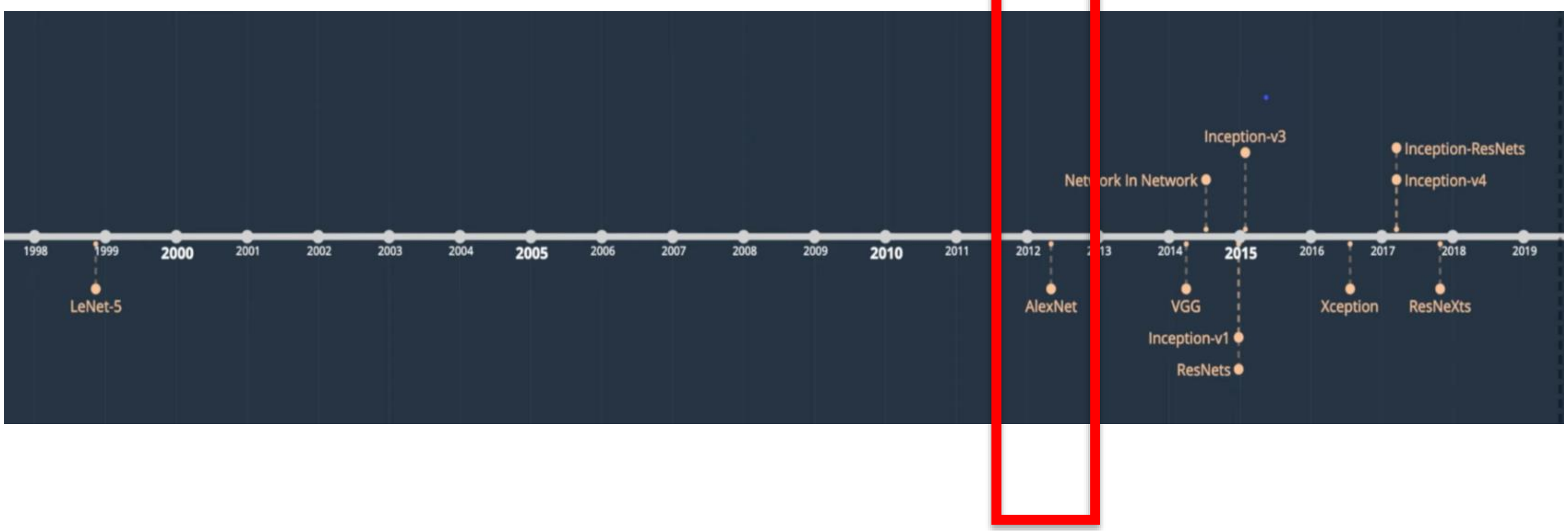
Why do we use the padding in CNN?

- zero-padding
  - It conserves the pixel information on the edge side
  - It can preserve the input's spatial size



AlexNet(2012)

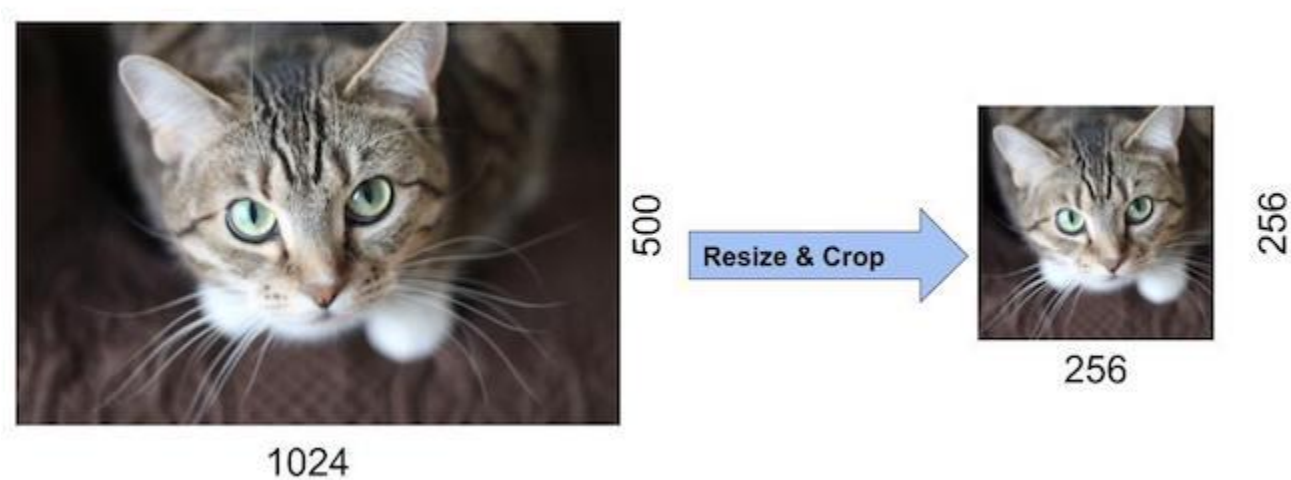
- ILSVRC



Contents

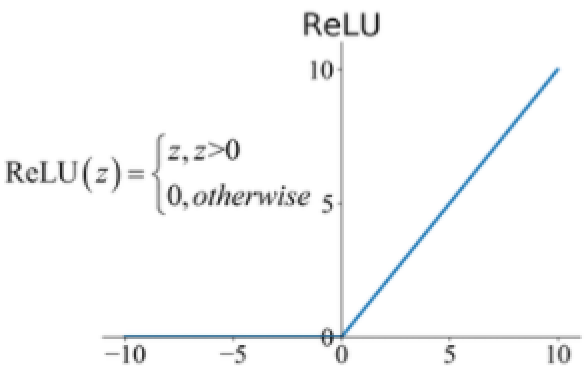
DataSet

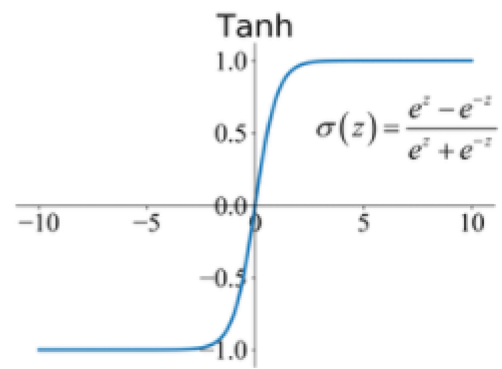
- An RGB image of size 256 x 256
  - If the input image is not 256 x 256 or 3-channel RGB
    - It needs to be converted to 256 x 256 before using it for training the network
    - It needs to be converted to an RGB image



Activation Function

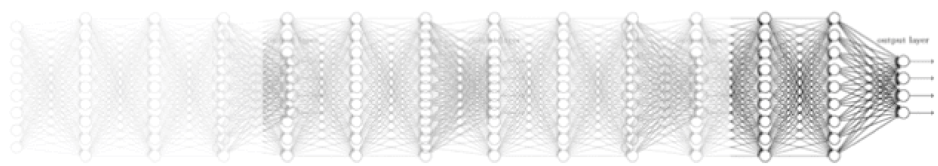
- ReLU
  - ReLUs train several times faster than their equivalents with tanh units.





Activation function

Vanishing gradient (NN winter2: 1986-2006)

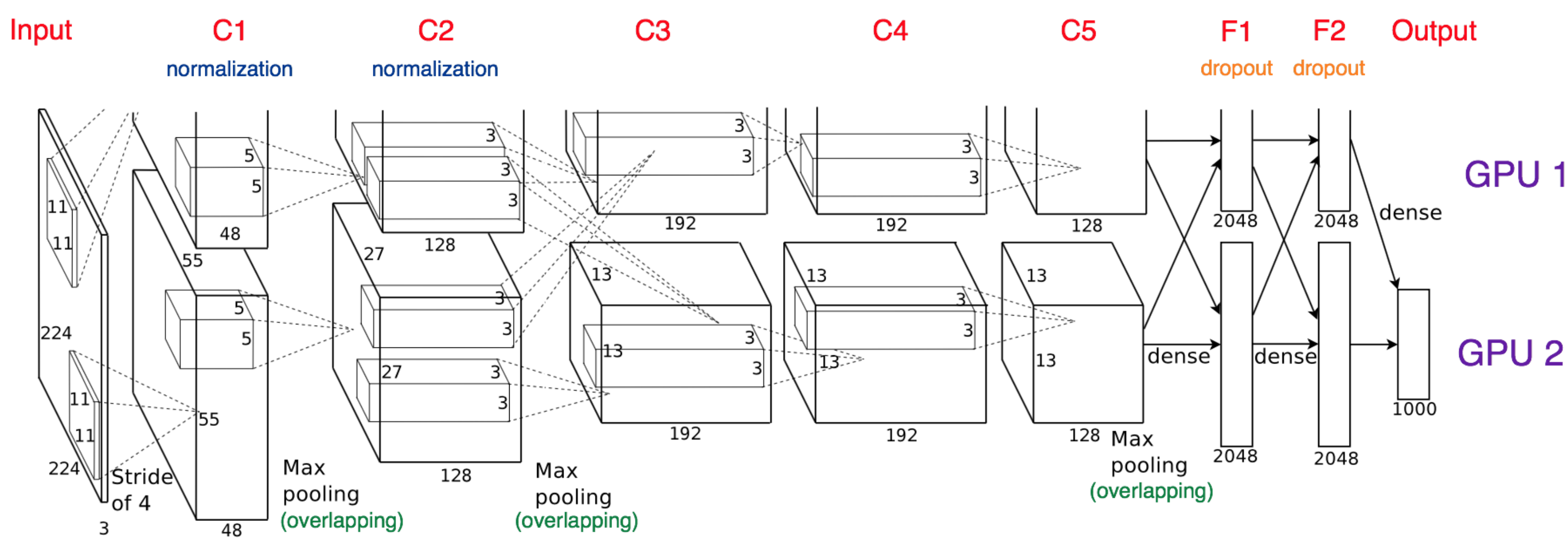


Activation functions on CIFAR-10

maxout	ReLU	VLReLU	tanh	Sigmoid
93.94	**92.11**	92.97	89.28	n/c
93.78	91.74	92.40	89.48	n/c
–	91.93	**93.09**	–	n/c
91.75	90.63	92.27	**89.82**	n/c
n/c†	90.91	92.43	89.54	n/c

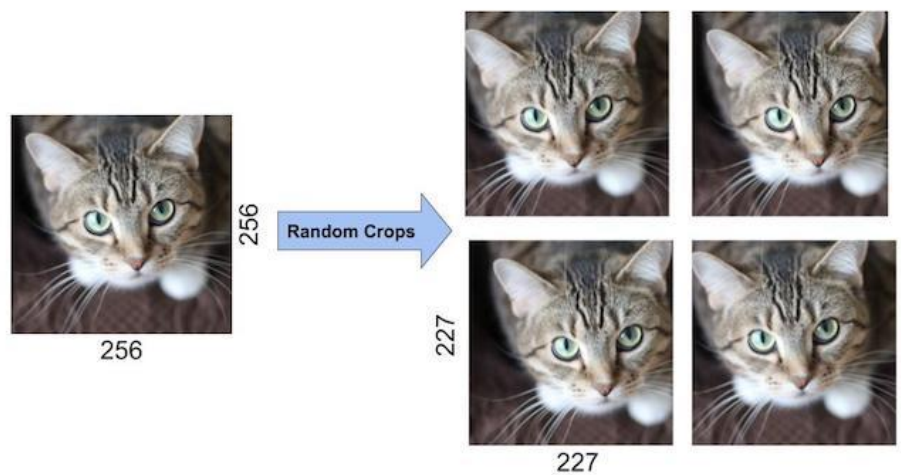
Training on Multiple GPUs

- GTX 580 3GB GPU
  - 5~6 days
    - 90 epochs
      - The results can be improved



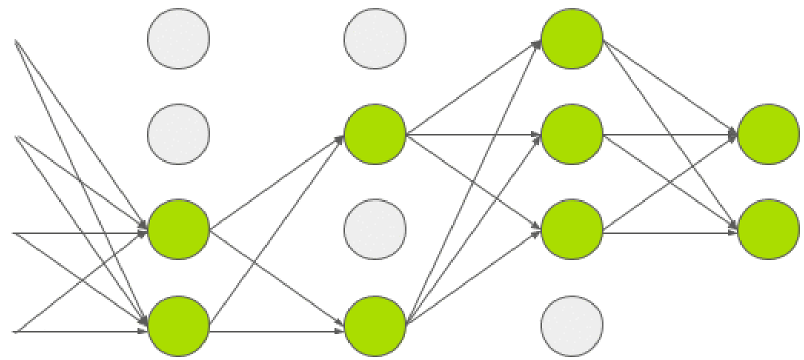
Reducing Overfitting

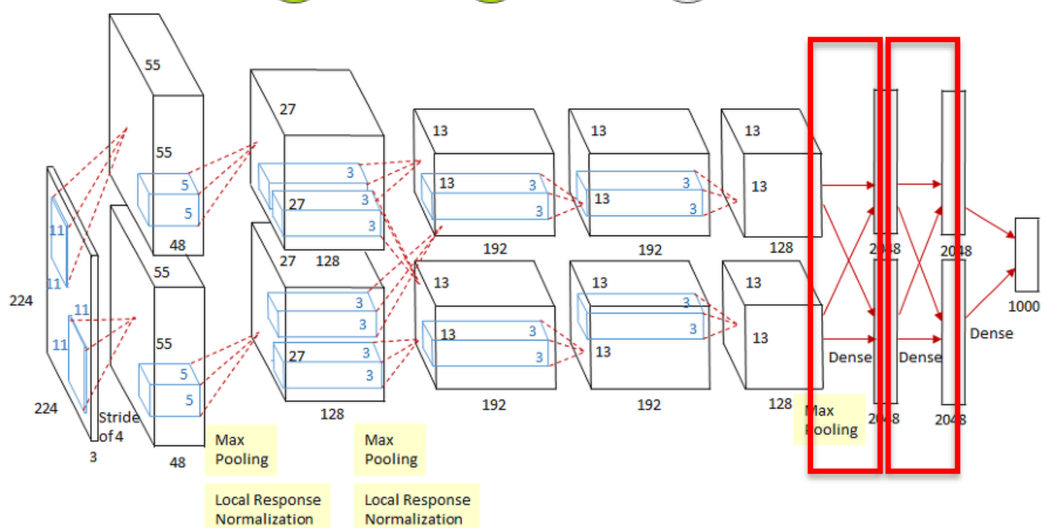
- Data Augmentation by Random crops
  - AlexNet input images are of size 227 by 227, which are randomly sampled from ImageNet's 256 by 256
    - The paper mentions the network inputs to be 224, but that is a mistake



Reducing Overfitting

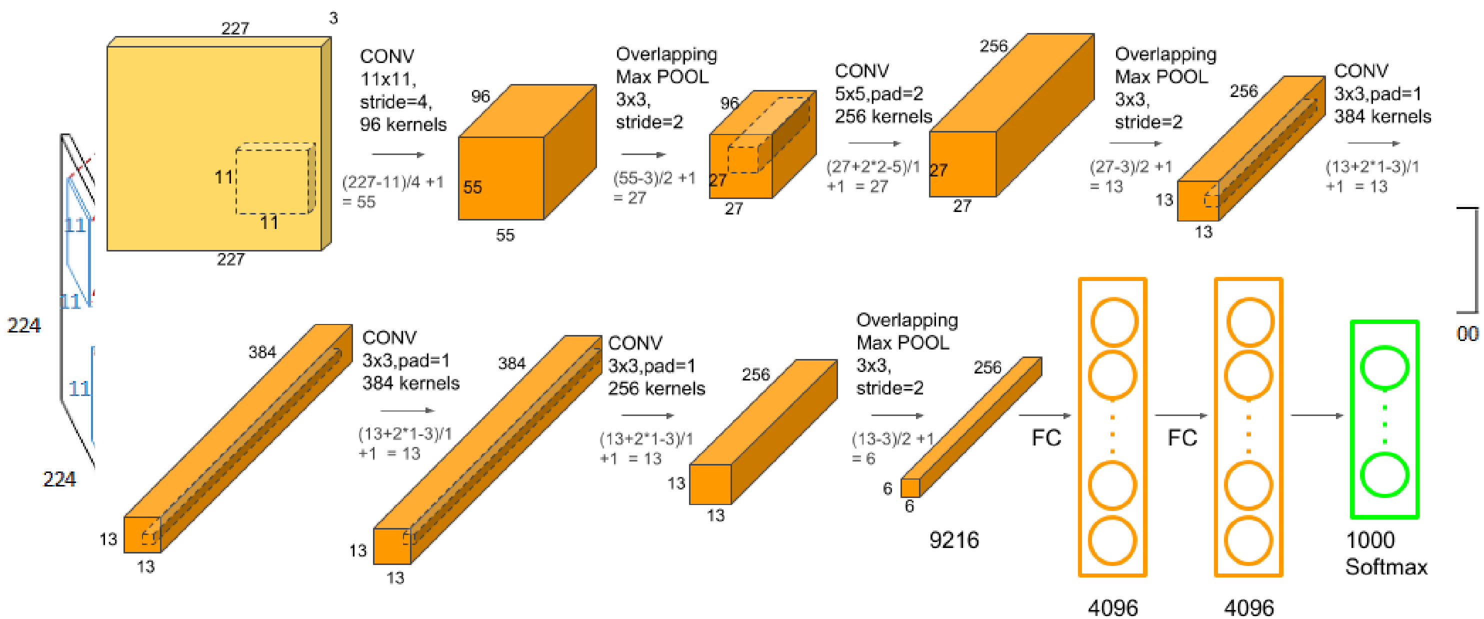
- Dropout





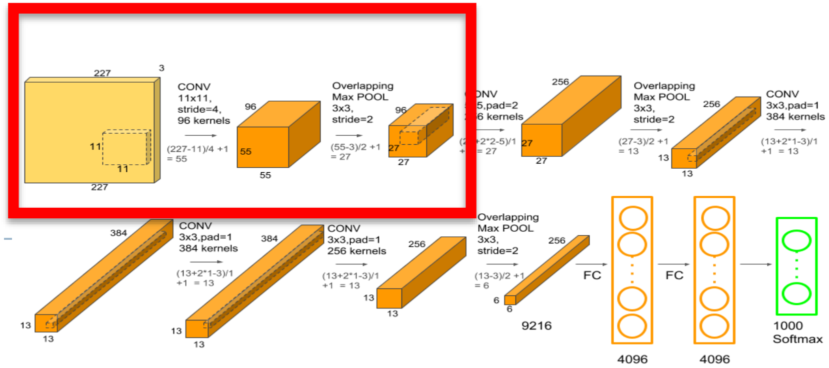
AlexNet Architecture

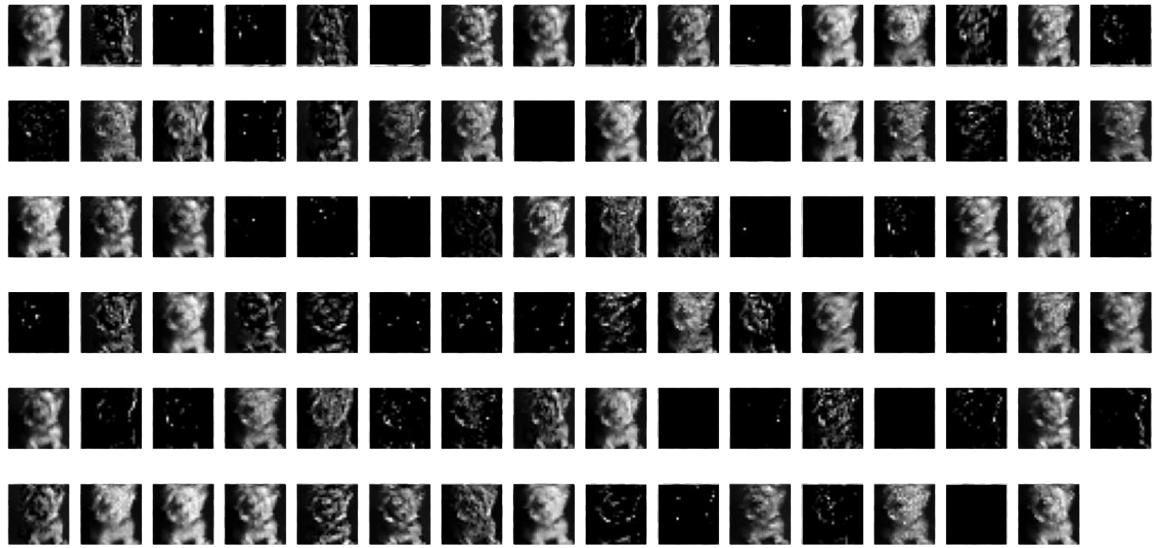
- First large scale convolutional neural network
- 5 Convolution layers
- 3 Fully connected layers



Feature Extractor

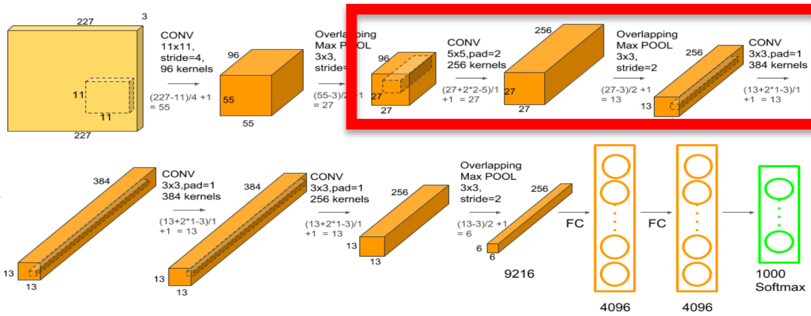
- 1st Convolutional Layer





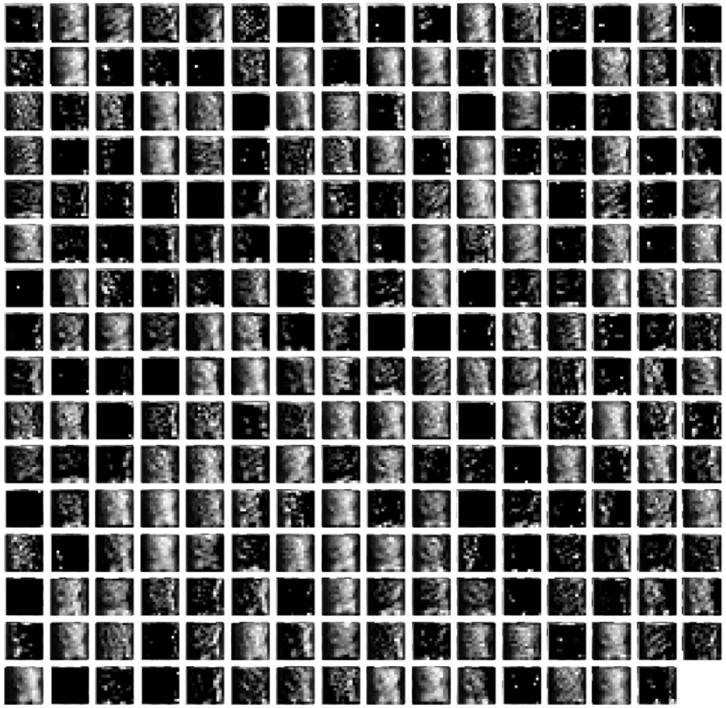
1 x 227 x 227 x 3(RGB)

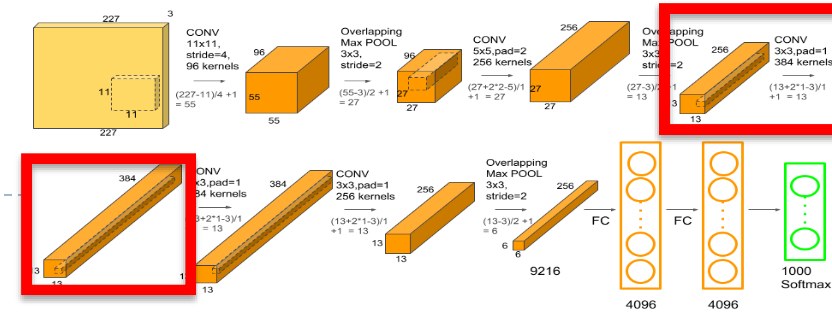
1 x 55 x 55 x 96



Feature Extractor

- 2nd Convolutional Layer

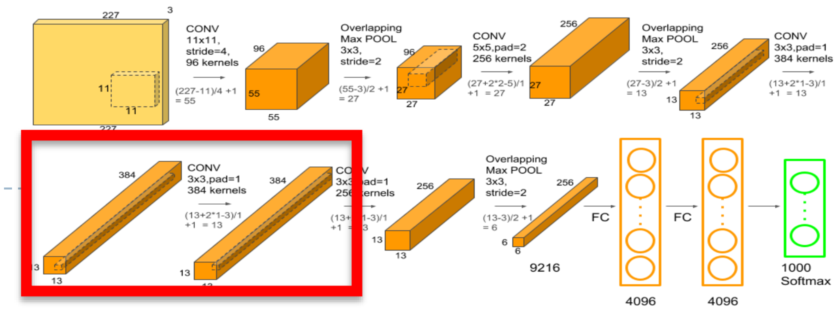




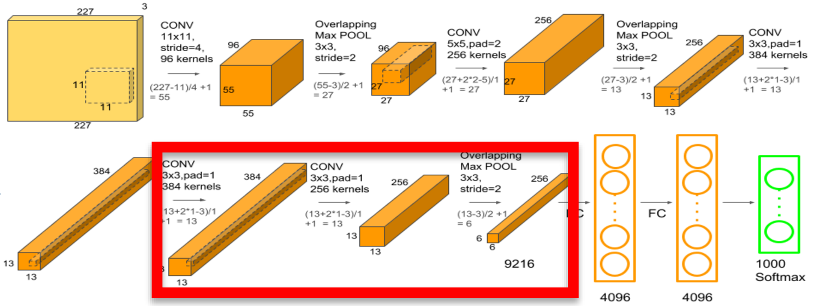
Feature Extractor

- 3rd Convolutional Layer

Feature Extractor

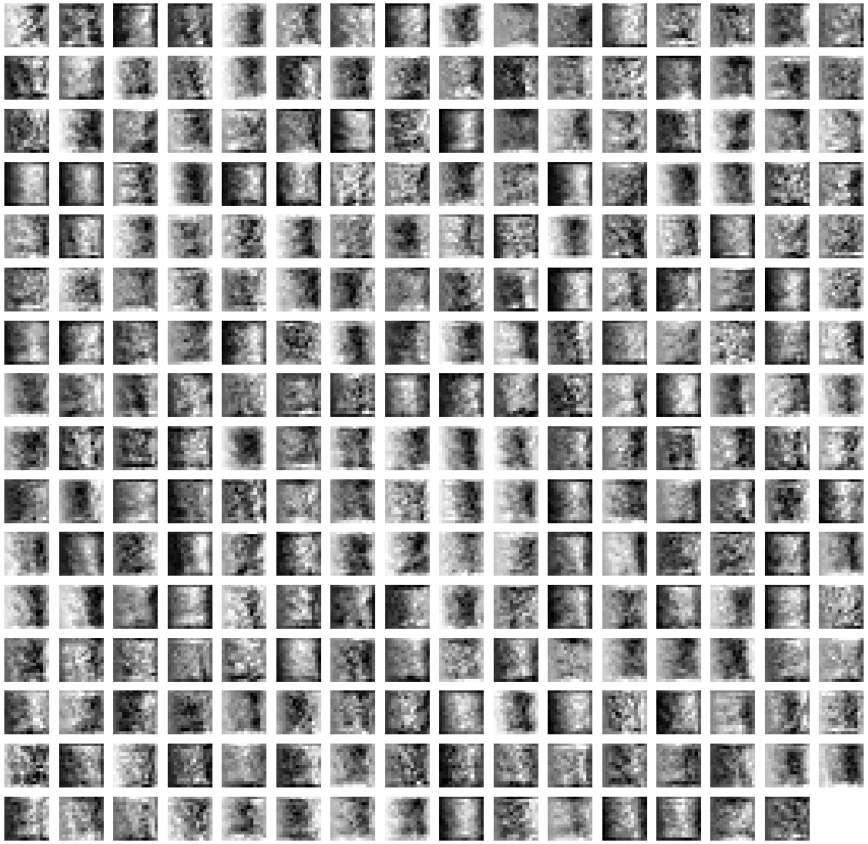


- 4^{th} Convolutional Layer

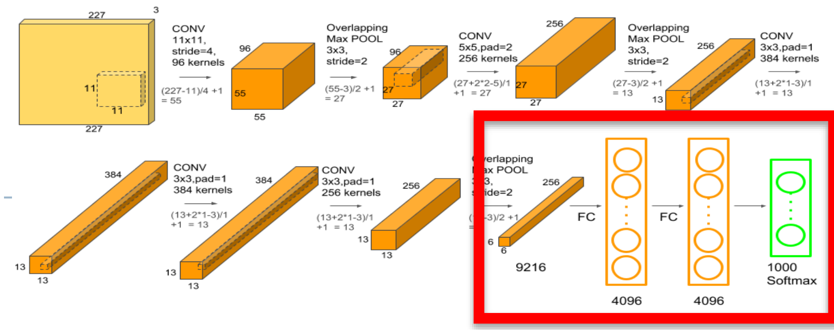


Feature Extractor

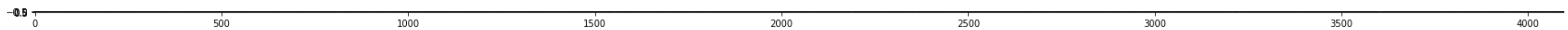
- 5^{th} Convolutional Layer



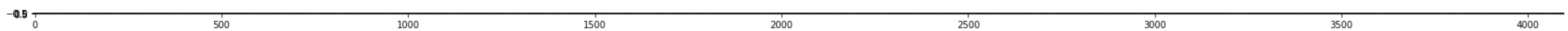
Classifier



- 1st Fully Connected Layer



- 2nd Fully Connected Layer



- Output Layer

AlexNet

Model

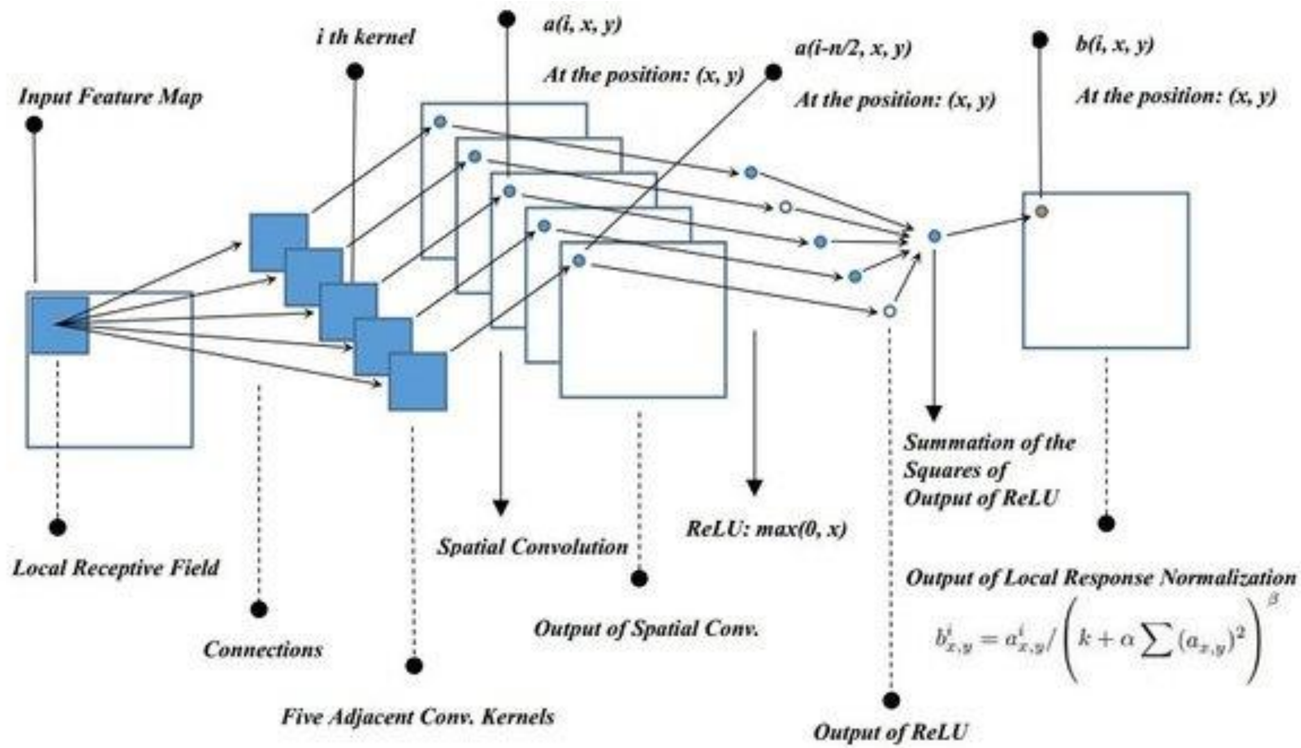
```
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=96, kernel_size=(11,11), strides=(4,4), activation='relu', input_shape=(227,227,3)),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Conv2D(filters=256, kernel_size=(5,5), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=256, kernel_size=(3,3), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
    keras.layers.Flatten(),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(10, activation='softmax')
])
```

Future work

- Data Augmentation
  - Principal component Analysis (PCA)

Future Work

- Local Response Normalization
  - ~: Batch Normalization



Local Response Normalization

- ReLUs do not require input normalization to prevent them from saturating
- However, Local Response Normalization aids generalization

Activity of a neuron by applying kernel i at position (x,y)

$$b_{x,y}^{i} = a_{x,y}^{i} \Big/ \left(k + \alpha \sum_{j=\max\left(0,i-\frac{n}{2}\right)}^{\min\left(N-1,i+\frac{n}{2}\right)} \left(a_{x,y}^{j}\right)^{2} \right)^{\beta}$$

$k = 2$
$n = 5$
$\alpha = 10^{-4}$
$\beta = 0.75$

sum runs over n "adjacent" kernel maps at the same spatial position

- Improvement:
  - top-1 error rate by 1.4%
  - top-5 error rate by 1.2%

Architecture

Local Response Normalization

- No need to input normalization with ReLUs.
- But still the following local normalization scheme helps generalization.

$$b^i_{x,y} = a^i_{x,y} \Bigg/ \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^\beta$$

Response-normalized activity

Activity of a neuron computed by applying kernel I at position (x,y) and then applying the ReLU nonlinearity

- Response normalization reduces top-1 and top-5 error rates by 1.4% and 1.2% , respectively.