SAN DIEGO STATE UNIVERSITY

CSRC

WWW.CSRC.SDSU.EDU

Image Classification using Convolutional Neural Networks

CS 596 – Machine Learning

Pradeep Singh

Agenda

Image Classification

- **Image classification:** A core task in Computer Vision

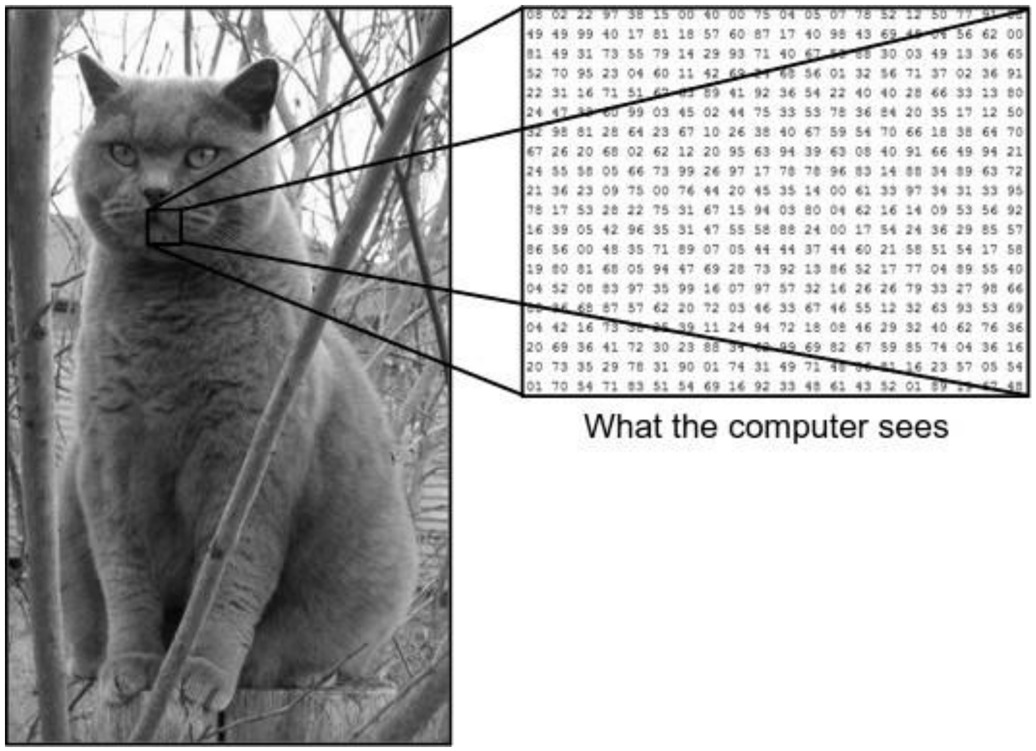
(assume given set of discrete labels)
{cat, dog, plane,...}

$\longrightarrow$ **Cat**

Image classification is the task of taking an input image and outputting a **class** or a **probability of classes** that best describes the image.

Image Classification

The semantic Gap: What we see vs. What computers see



Challenges:

1. Viewpoint Variation
2. Illumination
3. Deformation
4. Occlusion
5. Background Clutter
6. Intraclass Variation

For Humans this is an easy task, because our visual system and brain is hardwired for this sort of visual recognition tasks. But for machine this is a hard problem.

How to solve this problem ?

Traditional approaches:

Formulate algorithms of representation of image by encoding the existence of various features like corners, color-schemes, texture of image, etc. Also known as 'Hand-crafted features'.

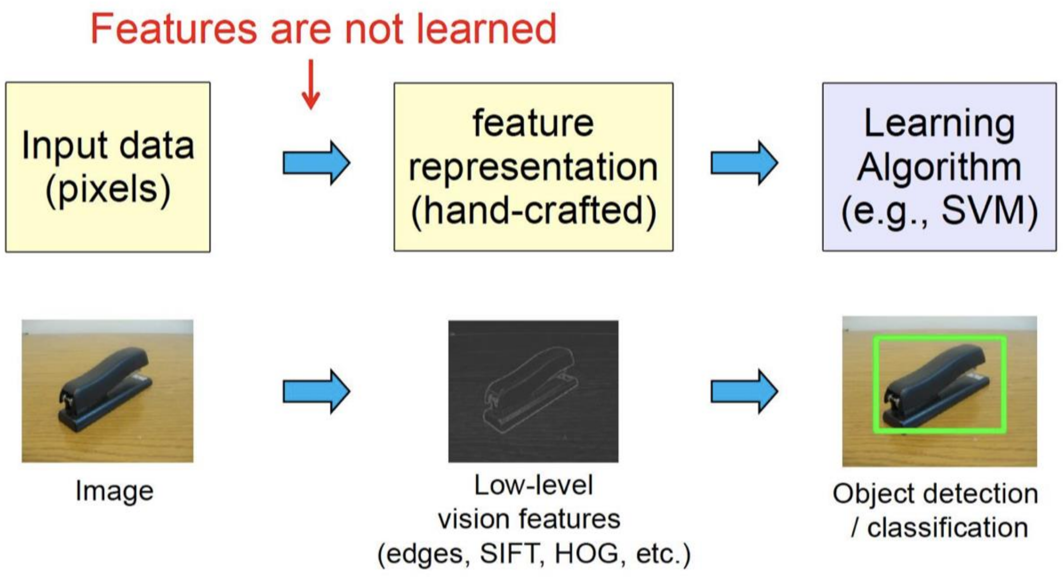

Features are not learned
Input data (pixels)
feature representation (hand-crafted)
Learning Algorithm (e.g., SVM)
Image
Low-level vision features (edges, SIFT, HOG, etc.)
Object detection / classification

Image: https://www.cs.tau.ac.il/~dcor

How to solve this problem ?

Data driven Approaches:

1. Formulate algorithms that extract features to discriminate objects; however, this time these features are learned via an automated procedure using deep-neural networks.

2. The general idea of deep-neural networks is to learn a denser and denser (more abstract) representation of the image as you proceed up the model.

3. Recent success in Computer vision is all because of the deep learning. So, I'll stick to deep learning.

Dataset

Fashion MNIST Dataset

Training Images: 60,000

Test Images: 10,000

Classes: 10

Current Best Accuracy: 97%

Defining Problem

Problem:

- Given an image, classify it as one of the 10 classes
- Multi-class classification
- Supervised learning
- Balanced class

Goal:

1) To build different neural network (CNNs, VGG etc.) models that can classify given images.
2) Comprehensive analysis of CNN models using various metrics.

Deep Learning

1. Type of machine learning, based on learning data representations.
2. Deep neural networks have a much higher parameter efficiency than shallow ones: they can model complex functions using fewer neurons.
3. Real world data is (often) structured in hierarchical way.

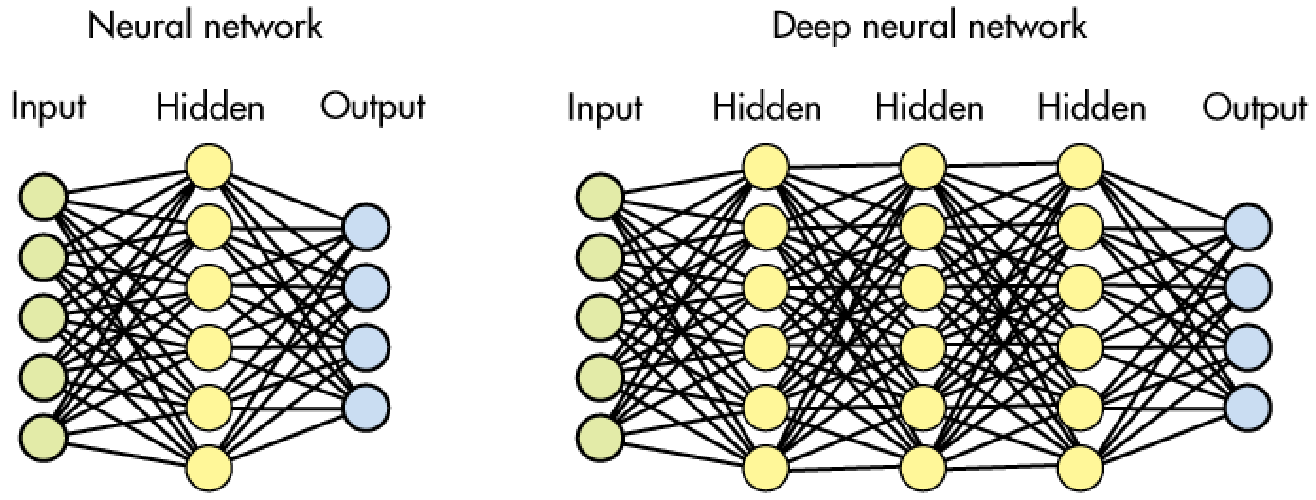
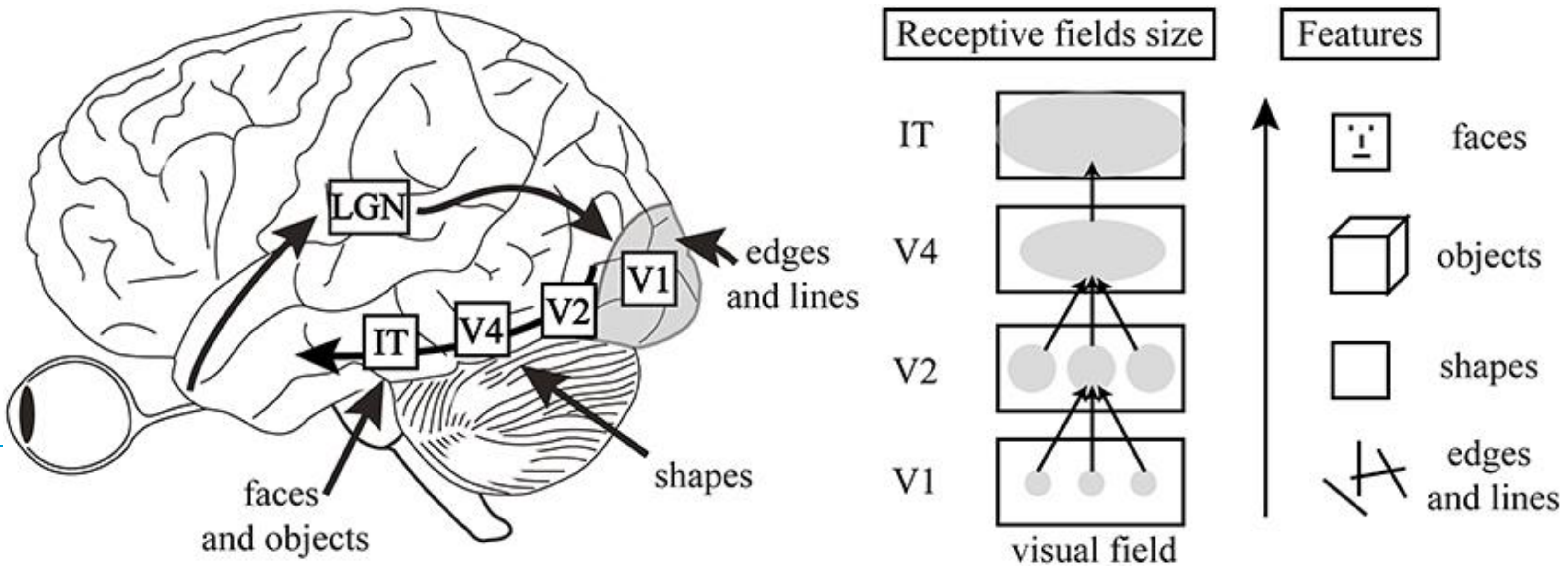


Image Ref: www.electronicdesign.com

Convolutional Neural Networks (CNNs)

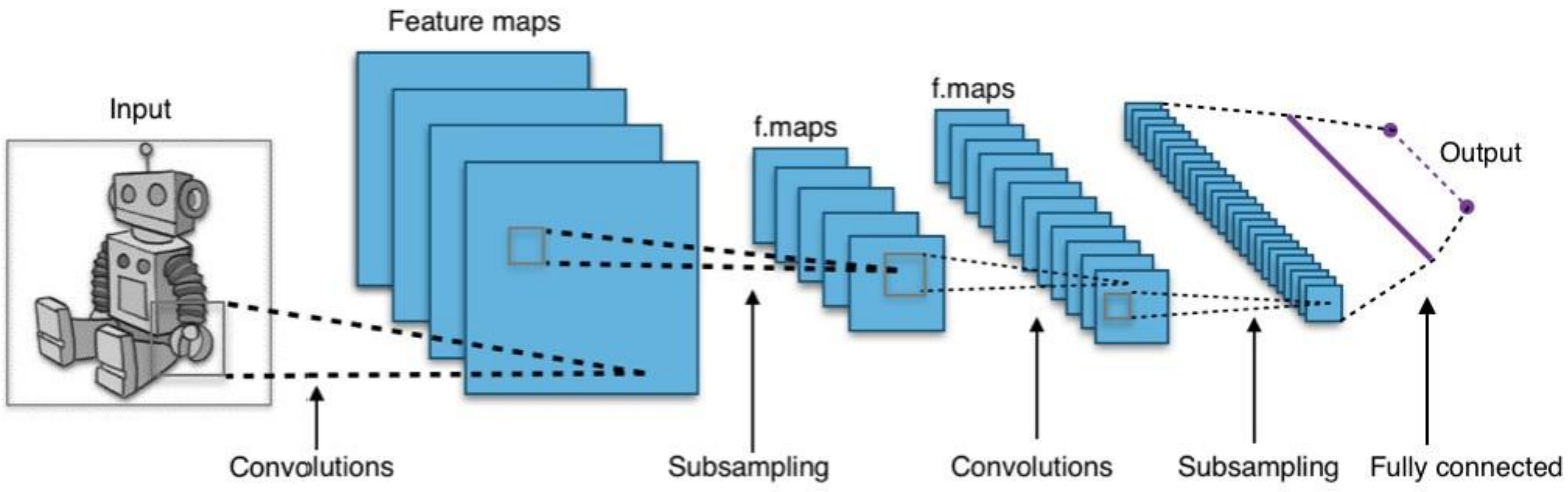
- CNNs takes inspiration from the visual cortex.
- The visual cortex has neurons that are sensitive to small local receptive fields i.e. they react only to visual stimuli located in a limited regions of the visual field.
- For eg: Some neurons react to **vertical** edges and some to **horizontal** edges.
- Having these neuronal cells in the visual cortex looking for specific characteristics is the basis behind CNNs.



Convolutional Neural Networks (CNNs)

Network Architecture

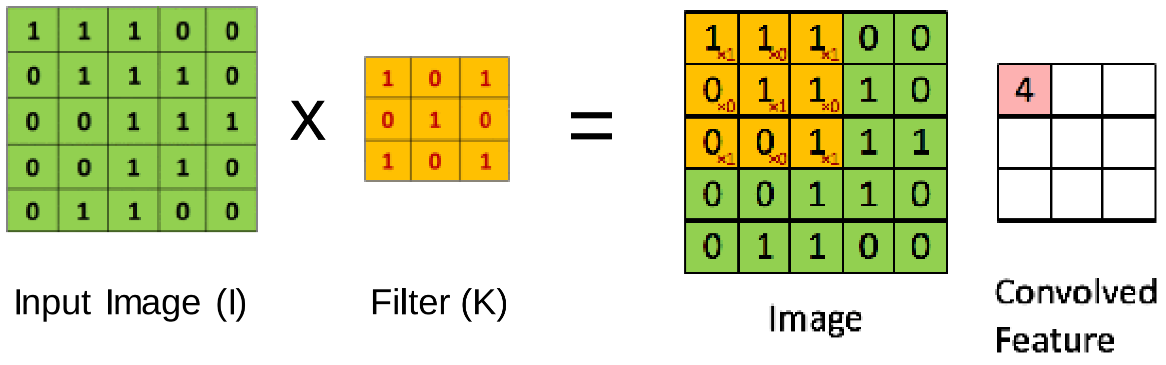
- Convolutional Layer,
- Pooling Layer,
- Fully Connected Layer,
- Activation Functions



Convolutional Layer in CNNs

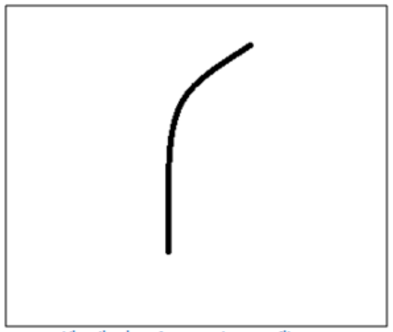
Convolution Operator

$$(I * K)_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{w} K_{ij} \cdot I_{x+i-1,y-j-1}$$



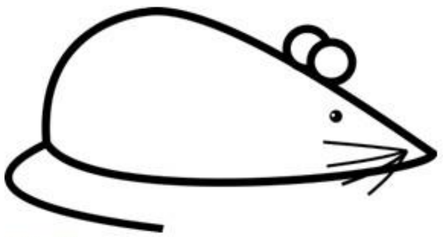
- Convolving the filter (K) with the image (I) i.e. “slide over the image spatially, computing dot products” is called as **Convolution**.
- The matrix formed by Convolution operation is called the **Feature Map**.
- CNN is a sequence of Convolution Layers, interspersed with activation functions.

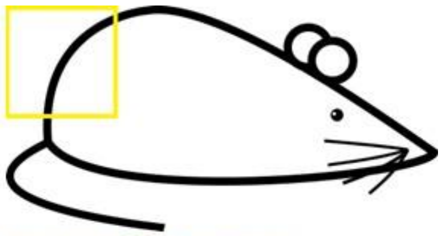
Convolutional Neural Networks (CNNs)

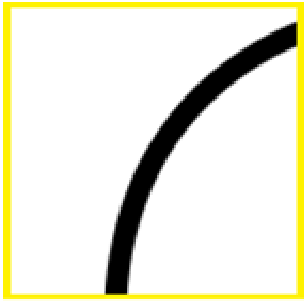

Visualization of a curve detector filter

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter


Original image


Visualization of the filter on the image


Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

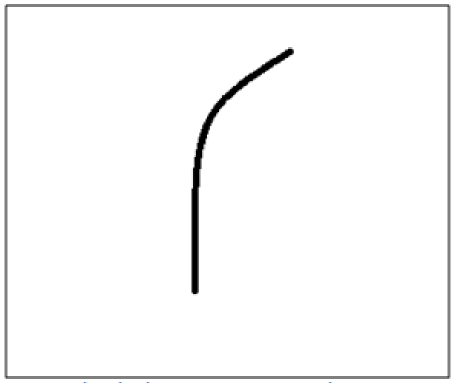
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

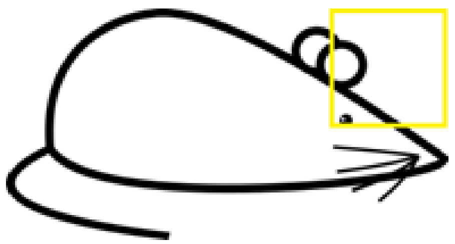
Convolutional Neural Networks (CNNs)



Visualization of a curve detector filter

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

$*$

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

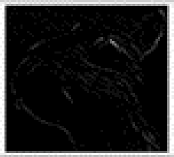
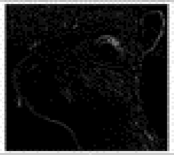
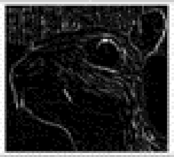
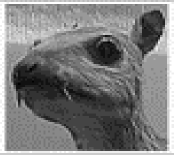
Multiplication and Summation = 0

Convolutional Layers

Convolution Operator

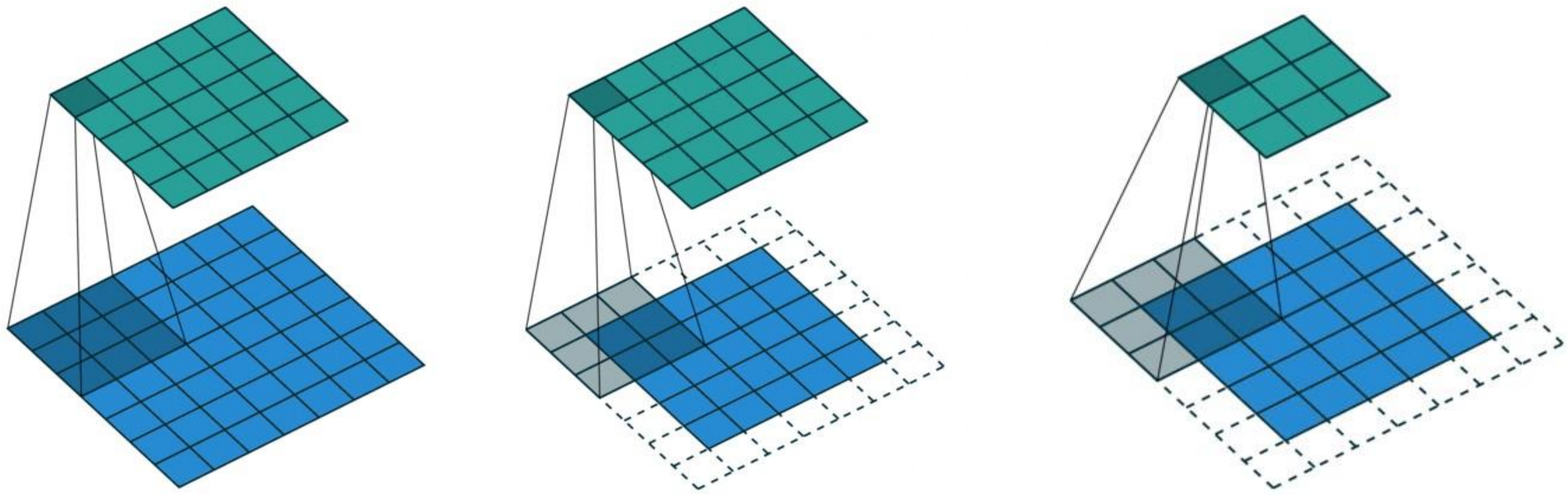
Different filters will produce different **feature maps** for the same input image.

Input Image

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Convolutional Layers

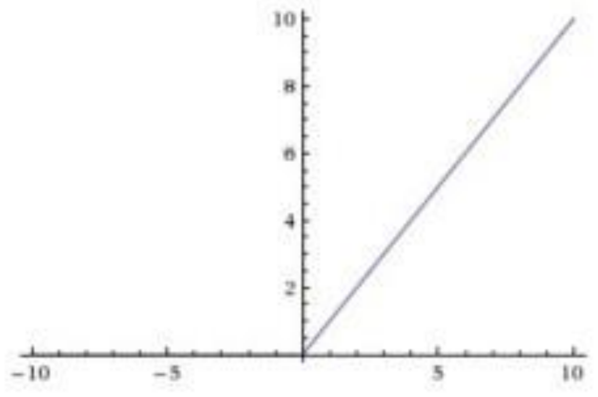
- CNN learns the values of these filters on its own during the training process.
- We need to specify parameter values such as number of filters, filter size, padding, stride before the training process.



Activation Functions (ReLU)

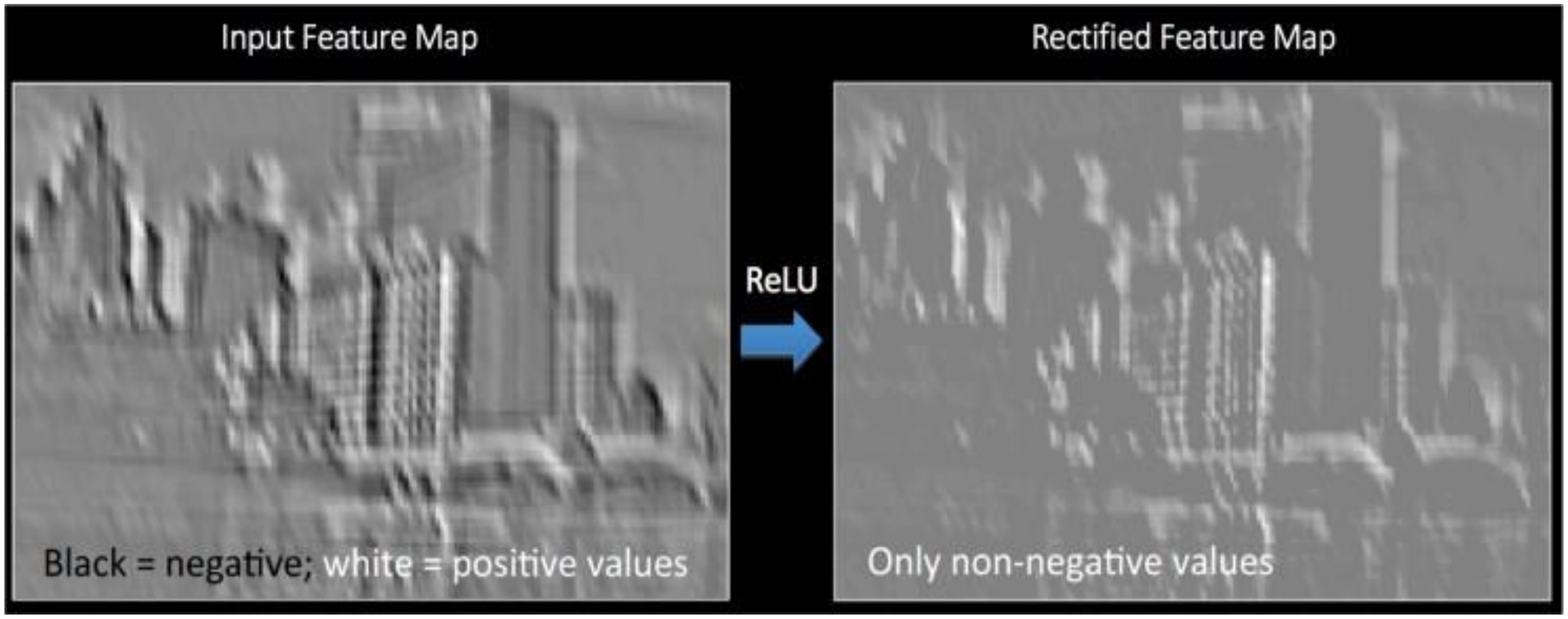
In order to introduce non-linearity in the network an additional operation called Rectified Linear Unit (ReLU) is used after every Convolution operation.

Output = Max(zero, Input)



ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero.

Activation Functions (ReLU)

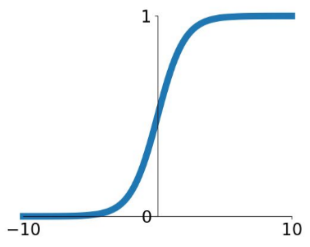


- Other non-linear functions such as ***tanh*** or ***sigmoid*** can also be used instead of ReLU.
- ReLU has been found to perform better in most situations.

Other Activation Functions

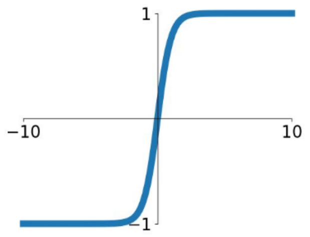
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



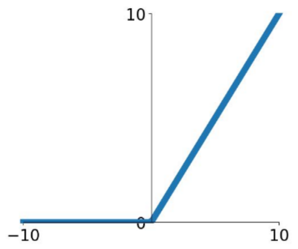
tanh

$\tanh(x)$



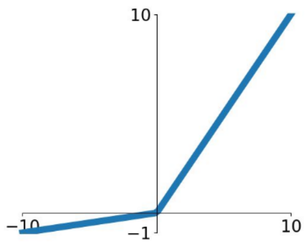
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

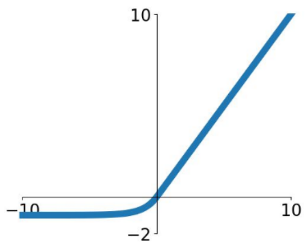


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

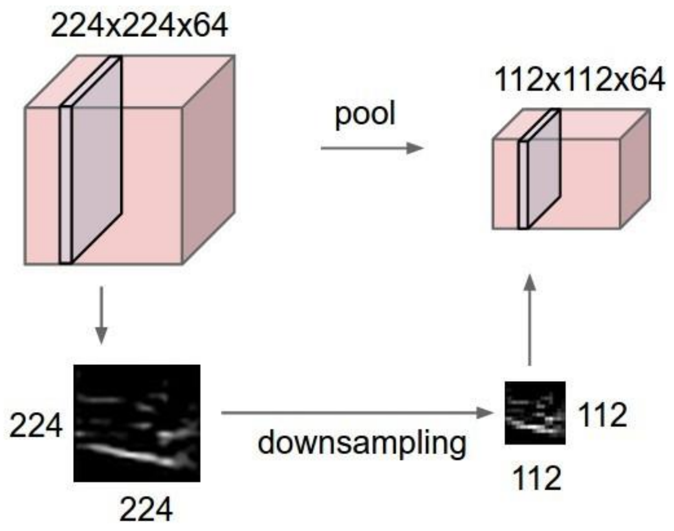
ELU

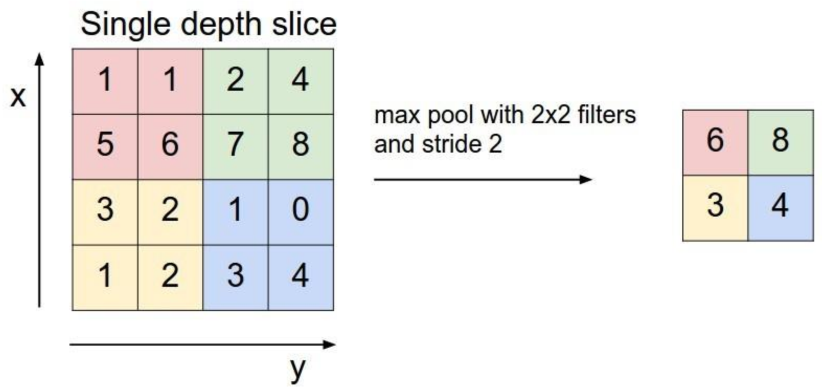
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Pooling Layers

- Pooling layer **down samples** the volume spatially, **independently** in **each depth** slice of the input.



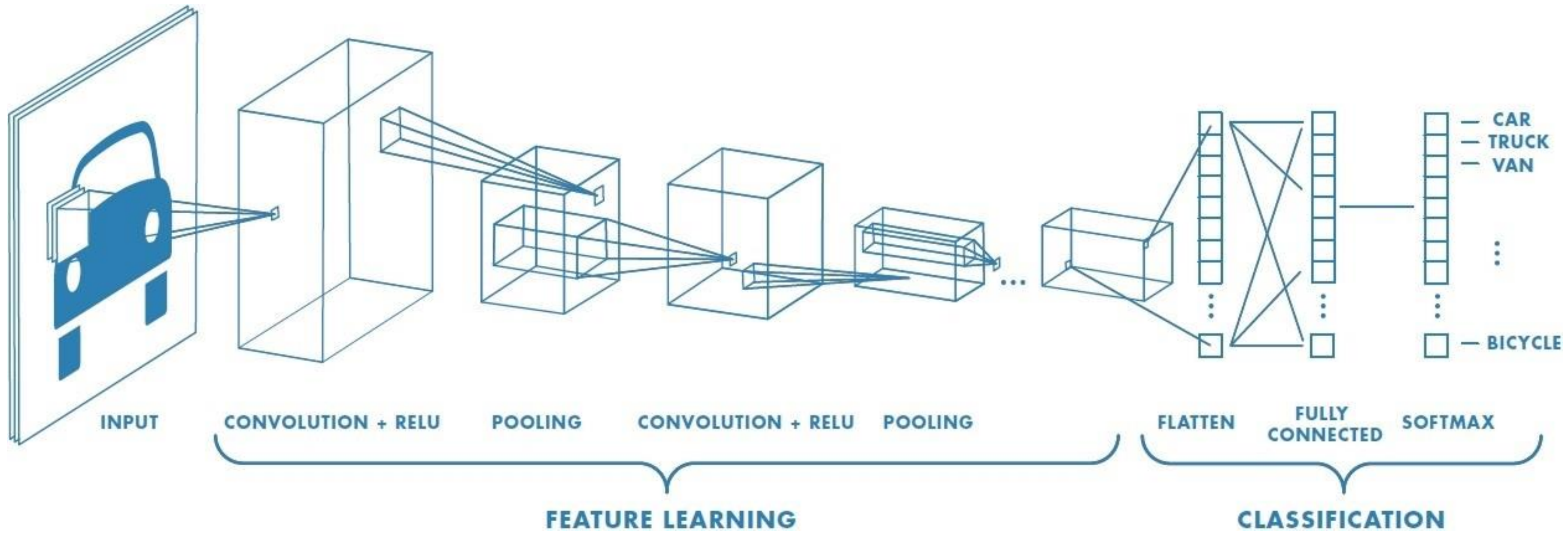


- Two types: Max Pooling and Average Pooling
- Intuition: Select most important features out of all features available.

Ref: CS231n http://cs231n.stanford.edu

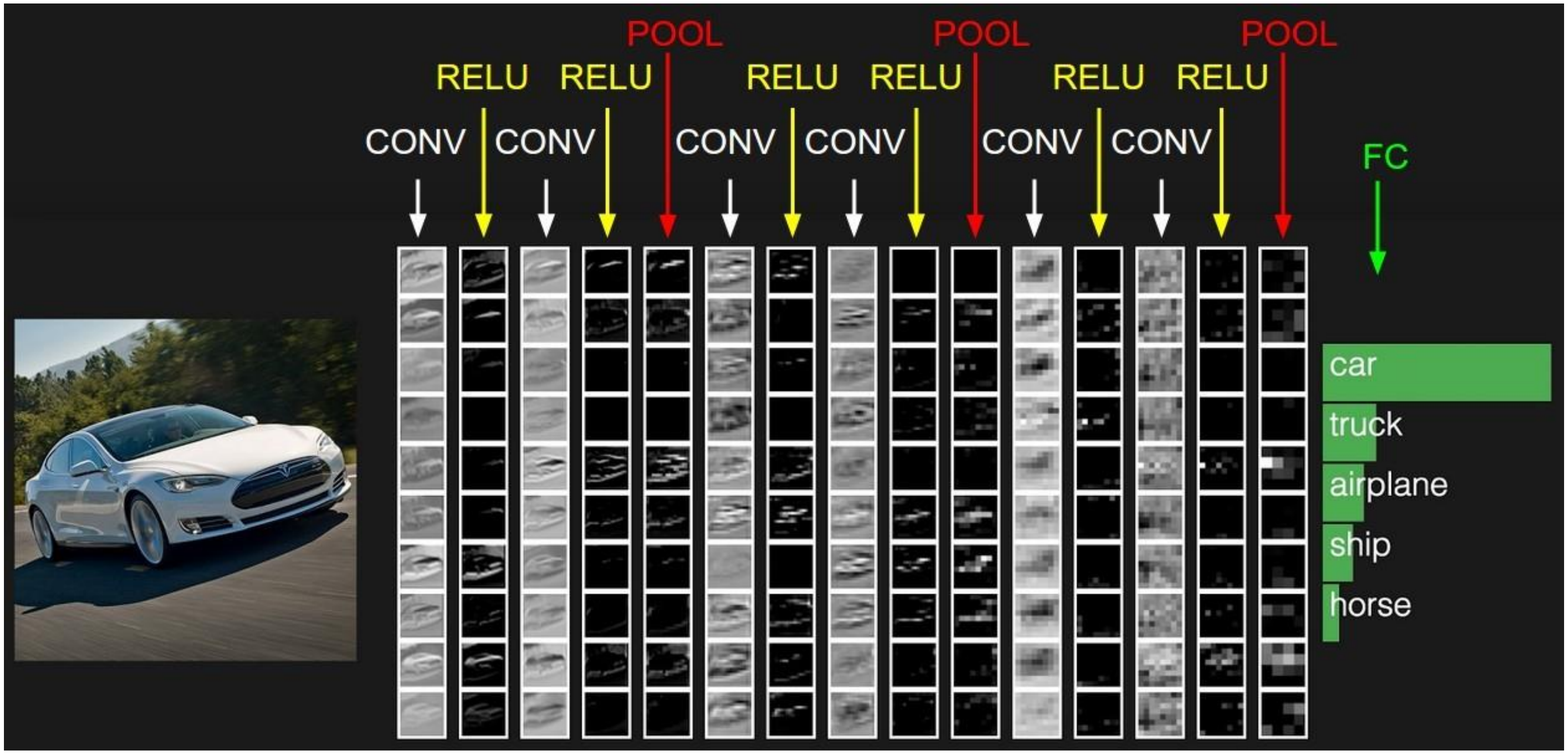
Full Connected Layers

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular neural networks.



Convolutional Neural Networks (CNNs)

Example: Input >> [[Conv >> ReLU] * 2 >> Pool] * 3 >> FC



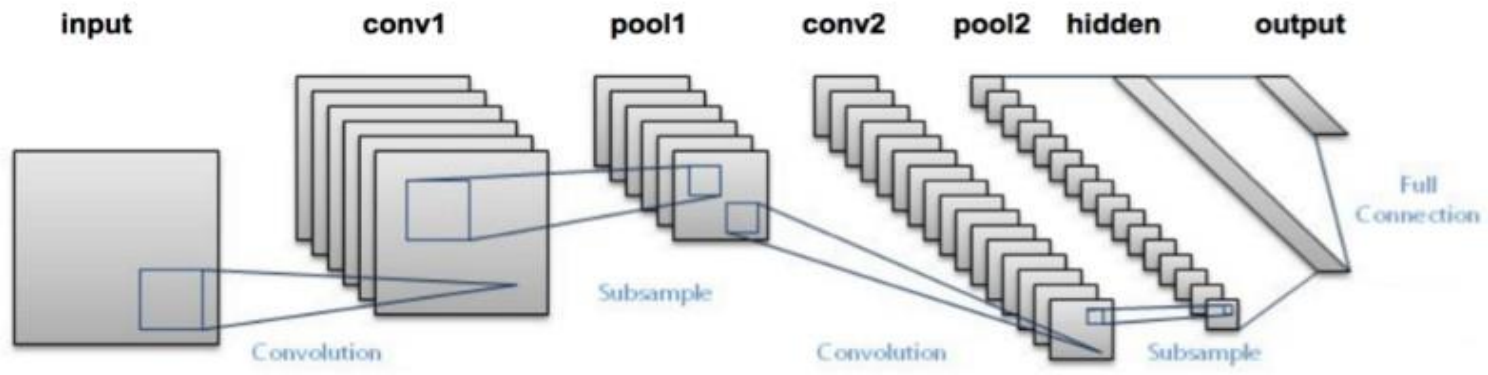
Ref: CS231n http://cs231n.stanford.edu

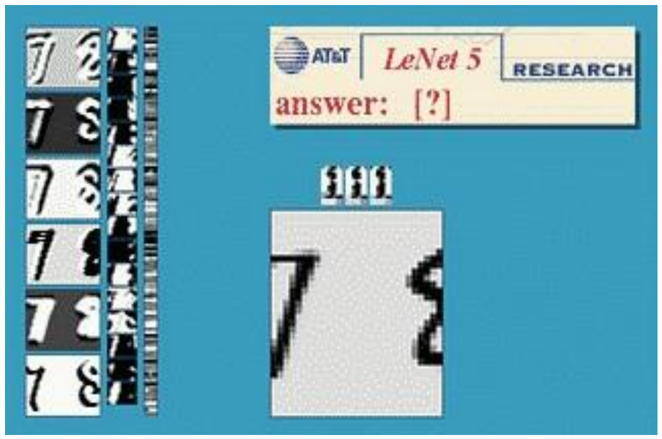
In Summary,

1. Dense layer learns global patterns, whereas convolutional layers learn local patterns.
2. Patterns they learn are translation invariant.
3. They can learn spatial hierarchies of patterns.
    - First layer will learn local patters such as edges.
    - Second layers will learn much larger patterns, made up of first layer.
    - And, so on.
4. Each layers may or may not have parameters. Types of layers:
    - Convolutional Layers
    - Activation Layers
    - Pooling Layers
    - Fully Connected Layers
5. In simple terms, CNN is a list of Layers that transforms the input volume into output volume (eg: class scores)

Successful CNN architectures

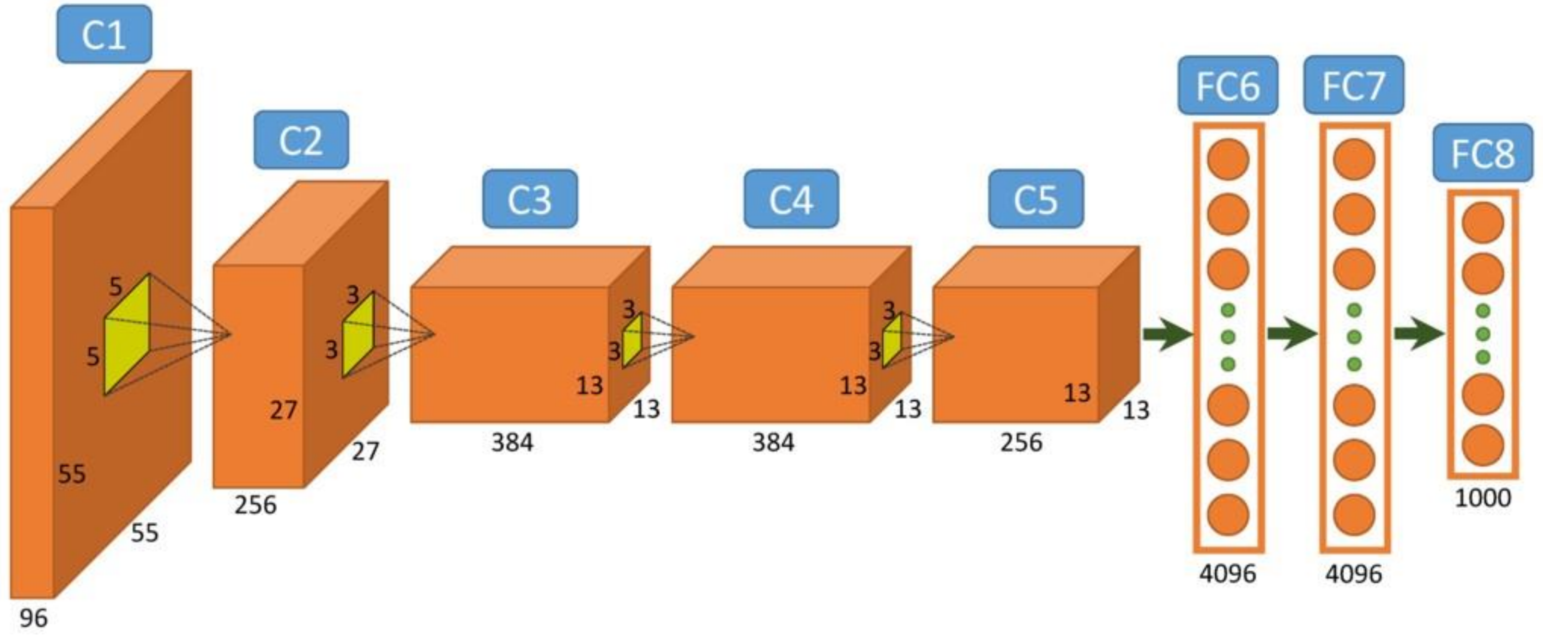
LeNet-5: The “first architecture” of CNNs developed in 1998 and was applied to document recognition tasks.





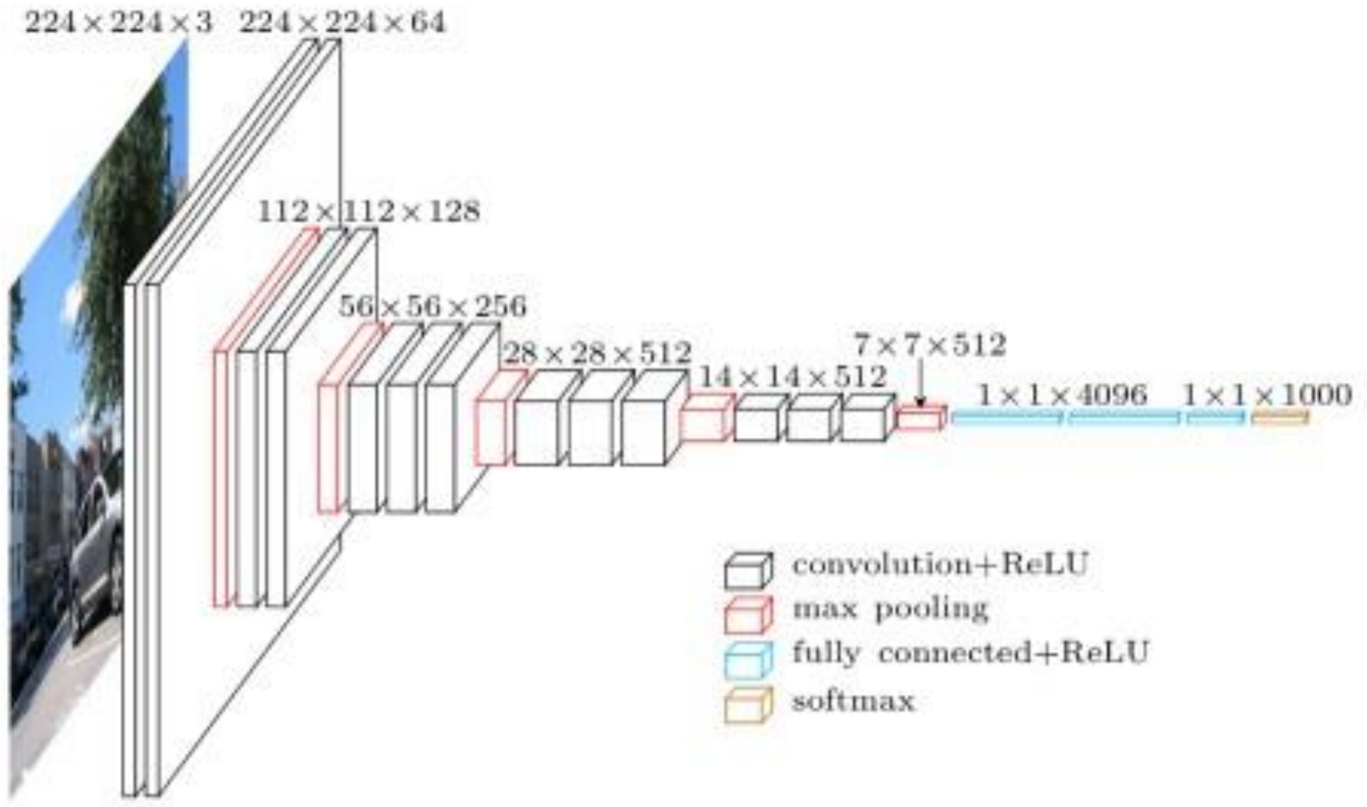
Successful CNN architectures

AlexNet: Famous for winning the **ImageNet** Large Scale Visual Recognition Challenge (**ILSVRC**) in 2012. Deep Learning breakthrough moment.



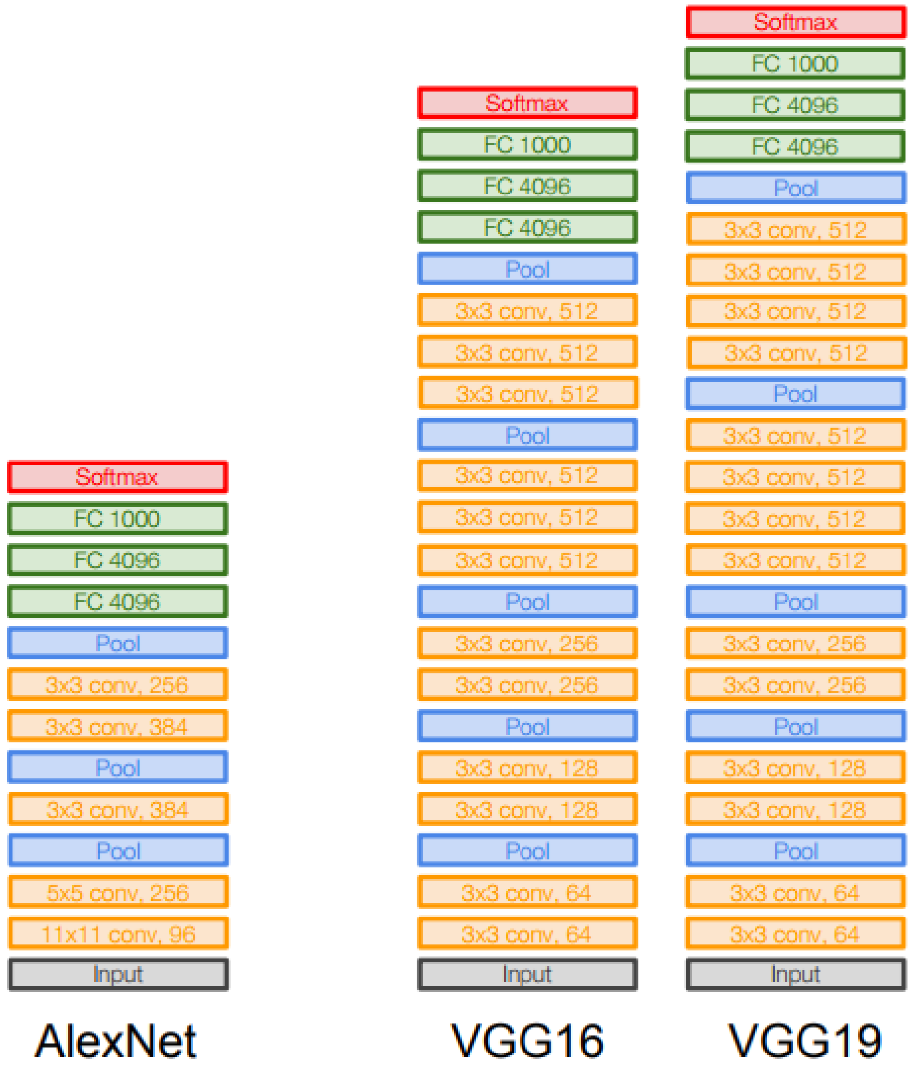
Successful CNN architectures

VGGNet



Successful CNN architectures

AlexNet vs. VGGNet (16 and 19)



Training

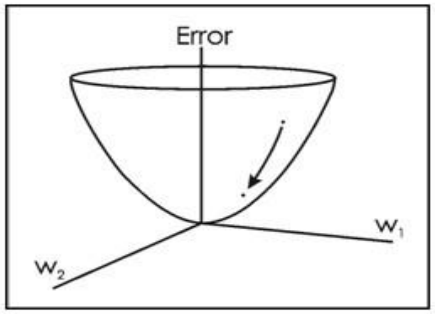
Backpropagation

- Algorithm to calculate all weights and biases.

- Cost Function

$$L_{total} = \sum \frac{1}{2}(target - output)^2$$

- Minimize gradient of the cost function.



- Weight Updates

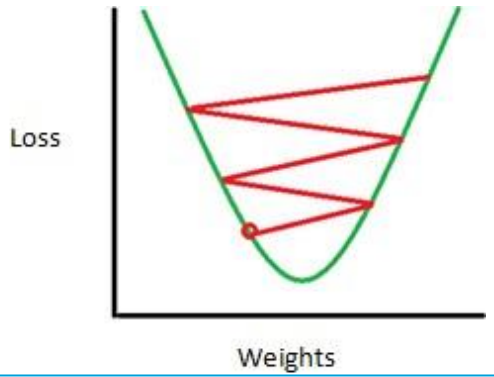
$$w = w_i - \eta \frac{dL}{dW}$$

w = Weight
w_i = Initial Weight
η = Learning Rate

Training

Backpropagation

- Learning Rate
  - Most important hyperparameter in deep learning.
  - A high learning rate means that bigger steps are taken in the weight updates. But it could result in steps that are too large and not precise enough to reach the optimal point.
  - Small learning rates means smaller steps, that could lot of time to train your models.



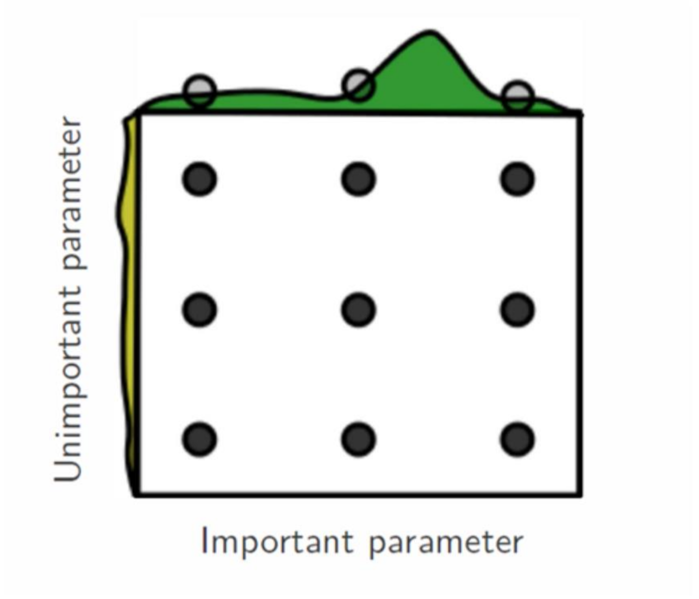
Hyperparameter Optimization

1. Special kind of parameters that cannot be directly learned from the regular training process.
2. Express “higher-level” properties of the model such as its complexity or how fast it should learn.
3. Fixed before the actual training process begins.
4. Examples: number of layers, learning rate, depth of a tree, size of filters, stride, pooling , etc.

Current Methods:

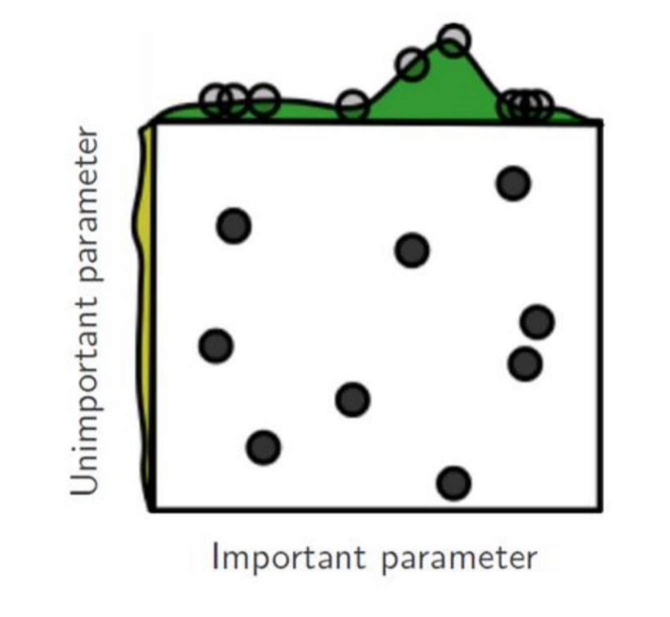
1. Grid Search
2. Manual Search (Grad student search)
3. Random Search (Current best)

Grid Search



Ref: Photo by James Bergstra (2012)

Random Search



Ref: Photo by James Bergstra (2012)

Manual Search

1. Humans manually pick and choose different values for different parameters and select the best one.
2. Humans have ability to learn from previous experience and mistake, which they use to evaluate their choices for parameters when tuning their models.
3. Least productive method.

Experiments

Approach:

- Build a small 2-layer neural network model
- Keep track of performance and errors
- Develop deeper models with intuition
- Do error analysis
- Try to improve the model/ performace
- Preprocessing – Data Augmentation
- More models
- Results

Experiments

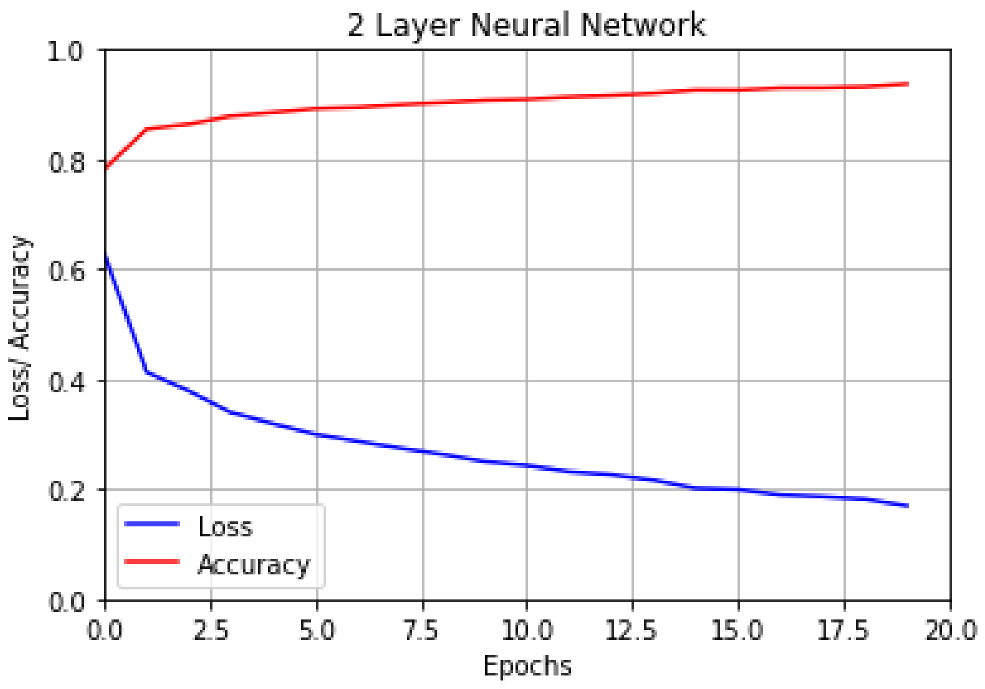
2 Layer Neural Network:

Hyper-parameters

Parameters	Value
Layers	2
Activation	ReLU
Learning Rate	0.001
Total Parameters	468,474

Evaluation Metrics

Data	Accuracy	Loss
Validation Set	0.93	0.16
Test Set	0.88	0.35



CNN with 1 Conv Layer:

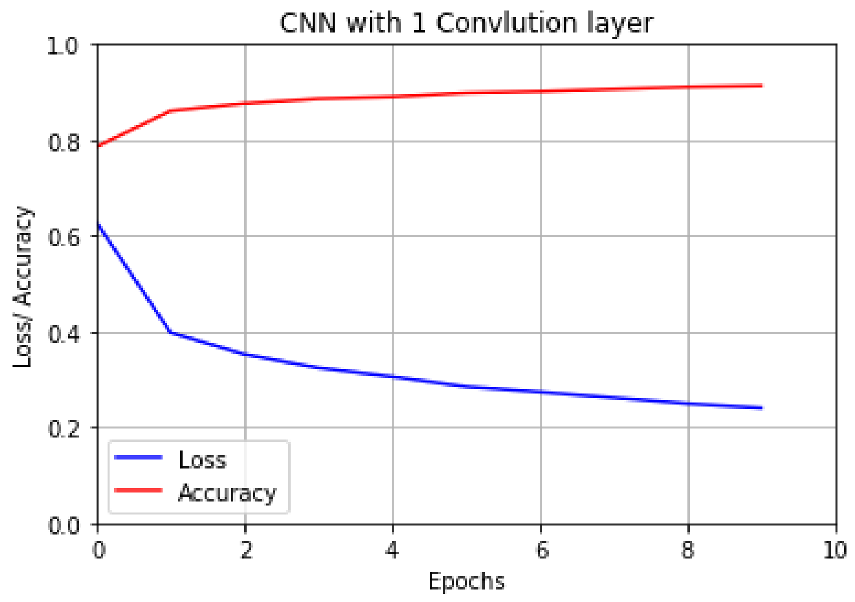
Hyper-parameters

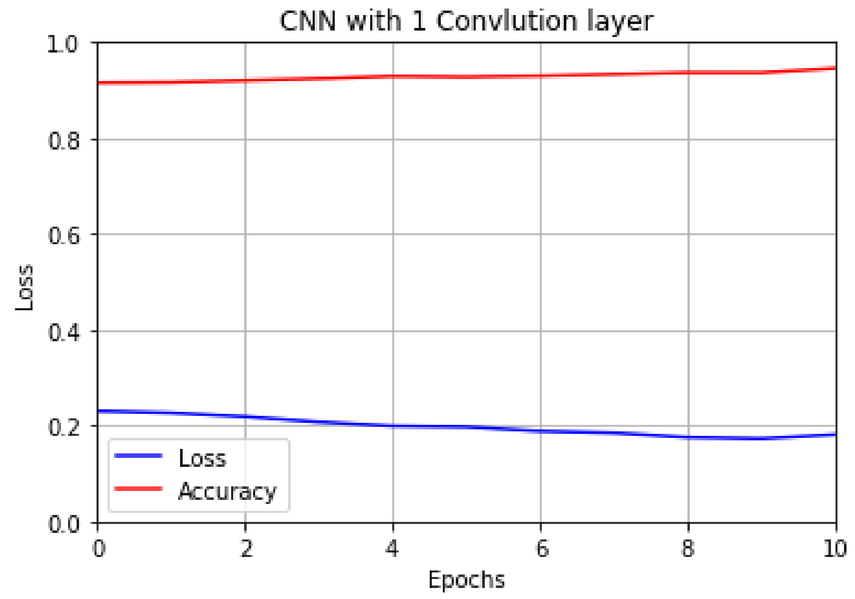
Parameters	Value
Conv Layer	1
Pooling Layer	1
FC Layer	1
Activation	ReLU
Learning Rate	0.001/ 0.0001
Total Parameters	693,962

Evaluation Metrics

Data	Accuracy	Loss
Validation Set	0.91	0.24
Test Set	0.91	0.25

Data	Accuracy	Loss
Validation Set	0.93	0.16
Test Set	0.88	0.35





CNN with 3 Conv Layer:

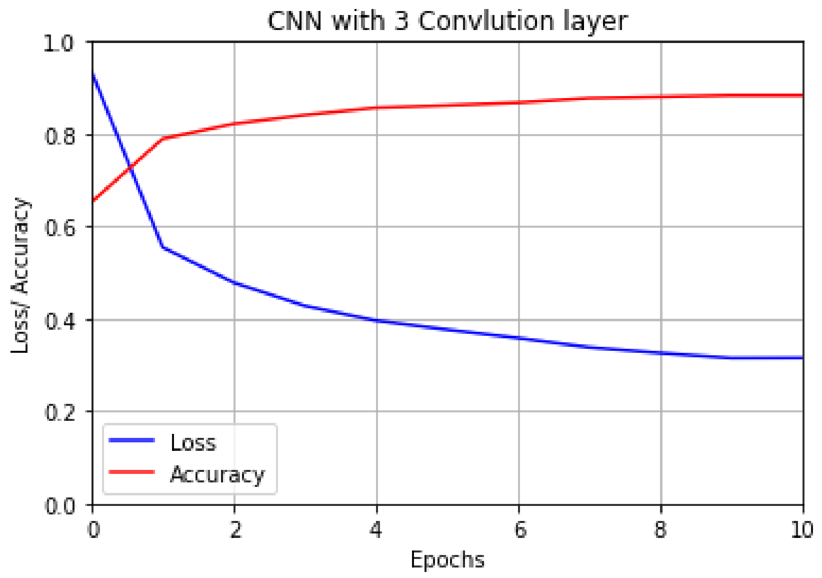
Hyper-parameters

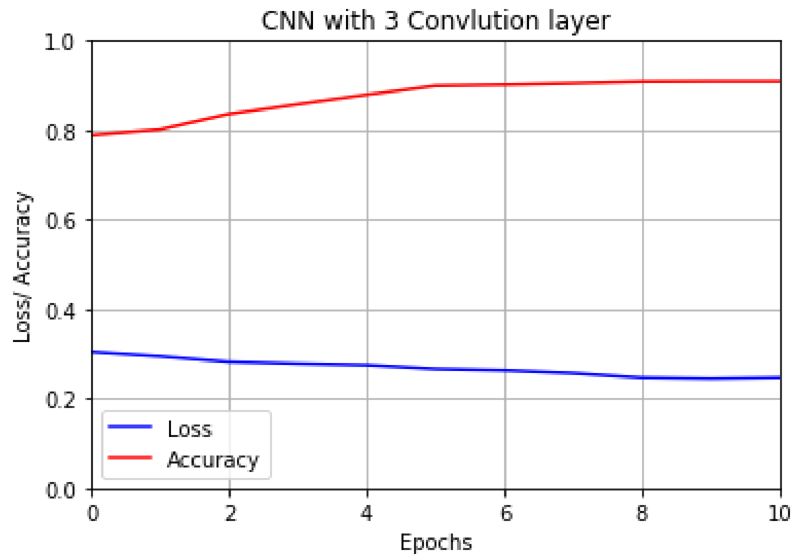
Parameters	Value
Conv Layer	3
Pooling Layer	2
FC Layer	1
Activation	ReLU
Learning Rate	0.001/ 0.0001
Total Parameters	241,546

Evaluation Metrics

Data	Accuracy	Loss
Validation Set	0.88	0.31
Test Set	0.88	0.35

Data	Accuracy	Loss
Validation Set	0.90	0.24
Test Set	0.89	0.25





CNN with 4 Conv Layer + Batch Normalization

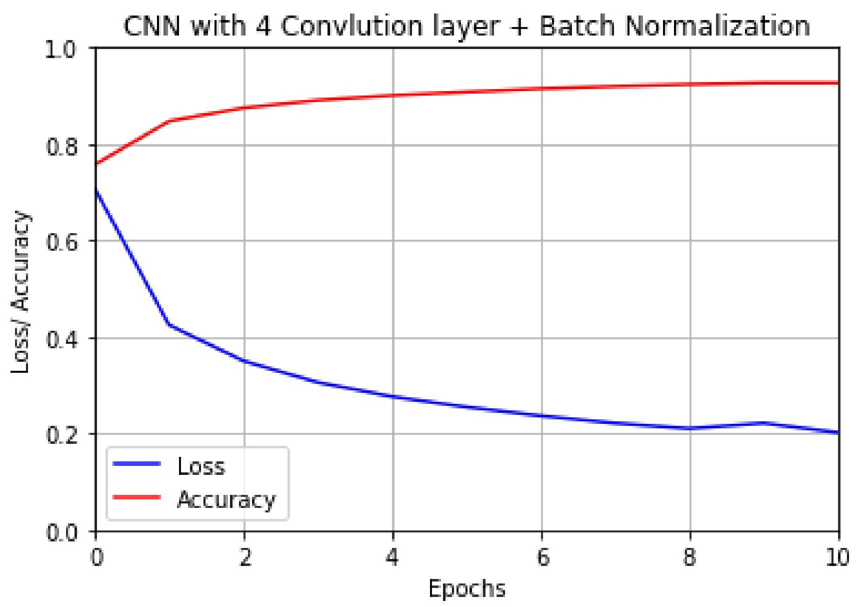
Hyper-parameters

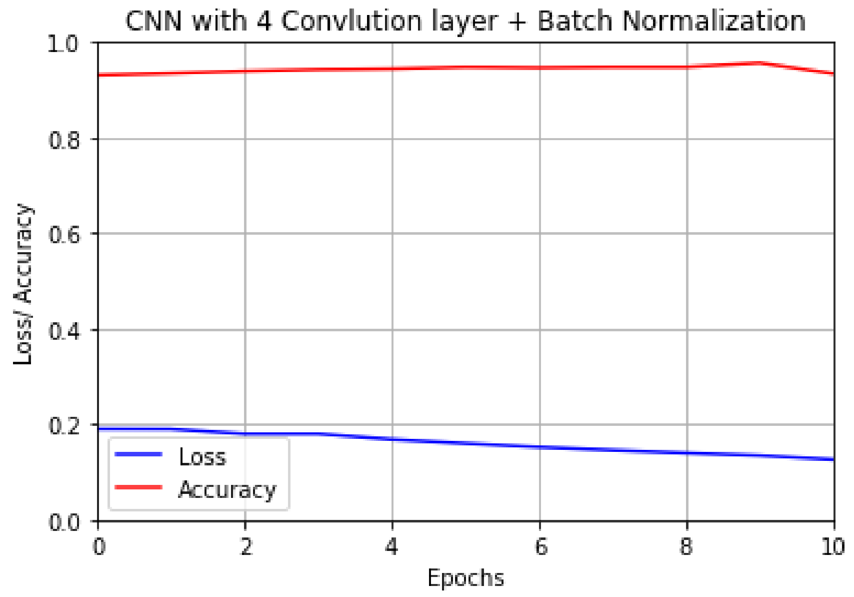
Parameters	Value
Conv Layer	4
Pooling Layer	1
FC Layer	1
Activation	ReLU
Learning Rate	0.001/ 0.0001
Total Parameters	241,546

Evaluation Metrics

Data	Accuracy	Loss
Validation Set	0.92	0.20
Test Set	0.92	0.21

Data	Accuracy	Loss
Validation Set	0.95	0.11
Test Set	0.93	0.22





VGG like model

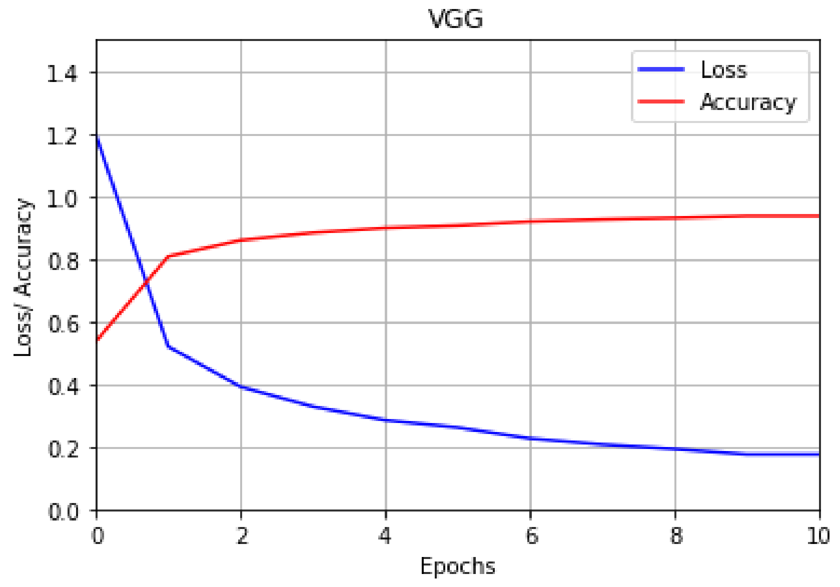
Hyper-parameters

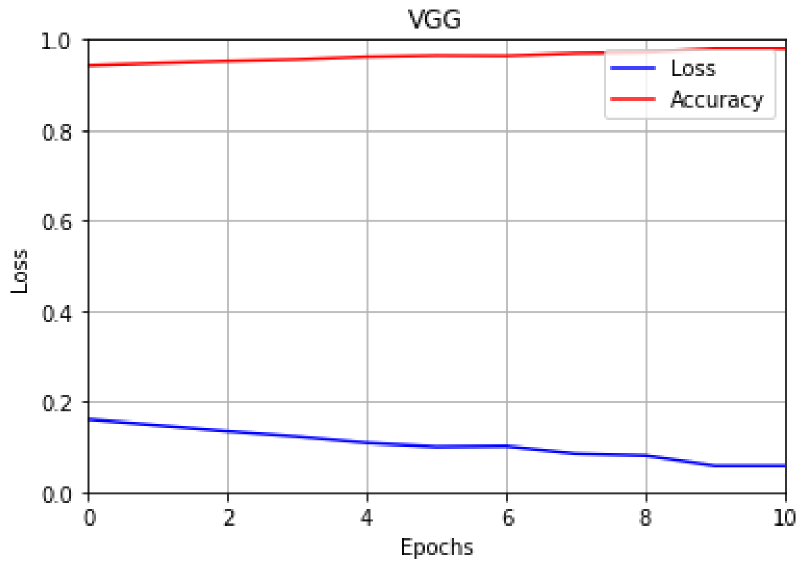
Parameters	Value
Conv Layer	9
Pooling Layer	4
FC Layer	1
Activation	ReLU
Learning Rate	0.001/ 0.0001
Total Parameters	241,546

Evaluation Metrics

Data	Accuracy	Loss
Validation Set	0.92	0.20
Test Set	0.92	0.21

Data	Accuracy	Loss
Validation Set	0.92	0.31
Test Set	0.91	0.33





VGG + Batch Normalization

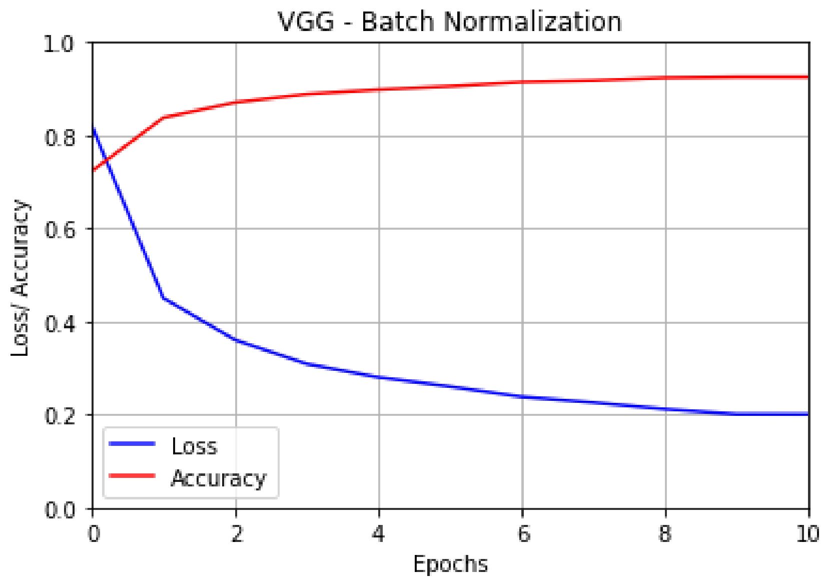
Hyper-parameters

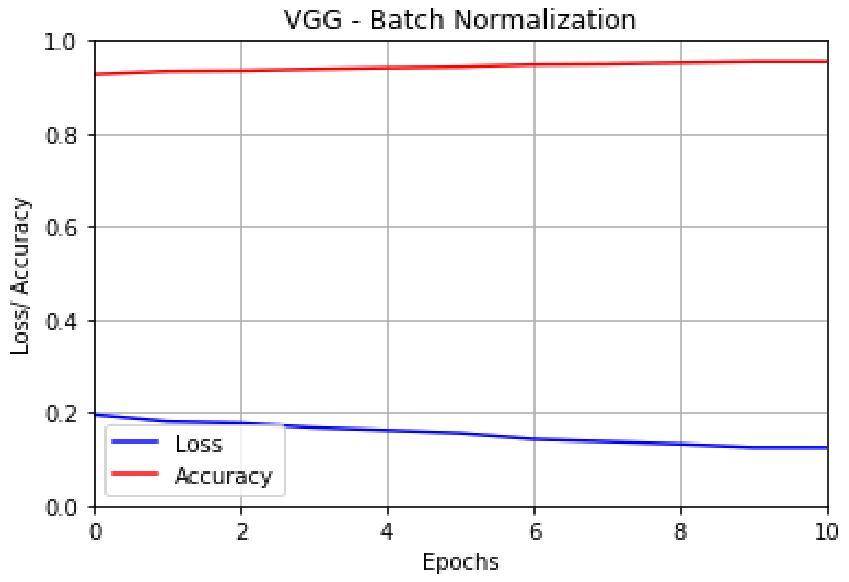
Parameters	Value
Conv Layer	9
Pooling Layer	4
FC Layer	1
Activation	ReLU
Learning Rate	0.001/ 0.0001
Total Parameters	241,546

Evaluation Metrics

Data	Accuracy	Loss
Validation Set	0.91	0.20
Test Set	0.92	0.20

Data	Accuracy	Loss
Validation Set	0.93	0.12
Test Set	0.93	0.11

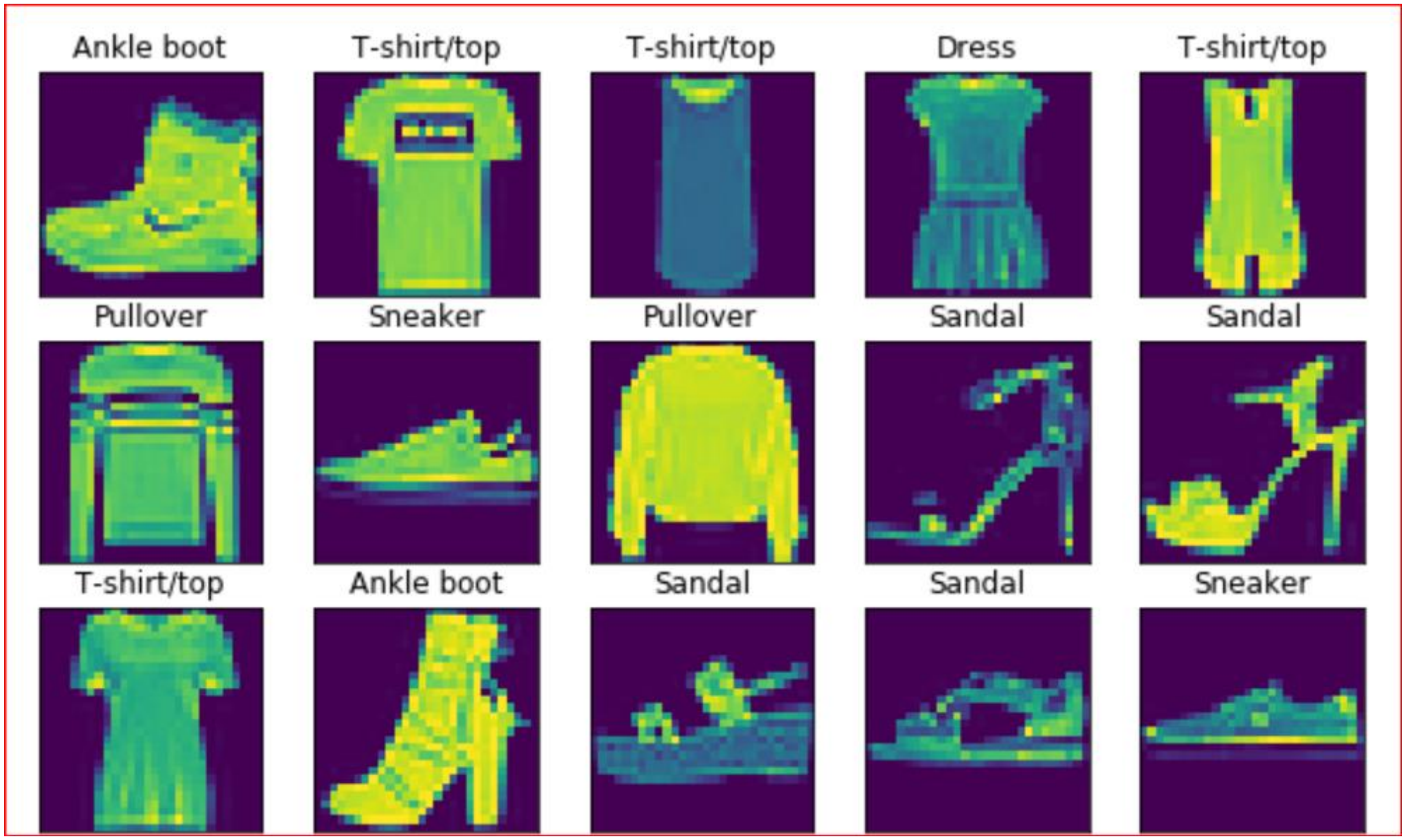




Error Analysis:

Confusion between similar looking classes

1. T-shirt vs Shirt vs Coat
2. Scandals vs Ankle Boots



Error Analysis:

What differentiate these similar looking classes?

- Low level and Mid-level features

How to make model learn these differences ?

- Data Augmentation
- Engineered Features
- More Data

Image Augmentation:

What ?

- Technique to artificially creates new training images through different ways of processing or combination of multiple processing.

Processing Tech ?

1. Random Rotation
2. Shifts
3. Shear
4. Flips
5. Contrast Adjustment

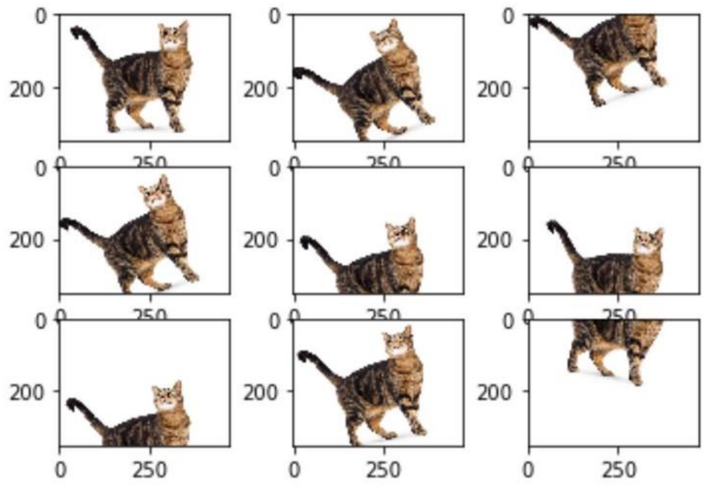
My *ImageDataGenerator* Settings:

Processing Tech.	Value
Rotation Range	8
Width Shift Range	0.08
Shear Range	0.3
Height Shift Range	0.08
Zoom Range	0.08

Image Augmentation:

How ?

- Image Data Generator API in Keras
- Generates batches of images with real-time data augmentation



Ref: https://towardsdatascience.com/image-augmentation-14a0aafd0498

More Experiments: with Data Augmentation

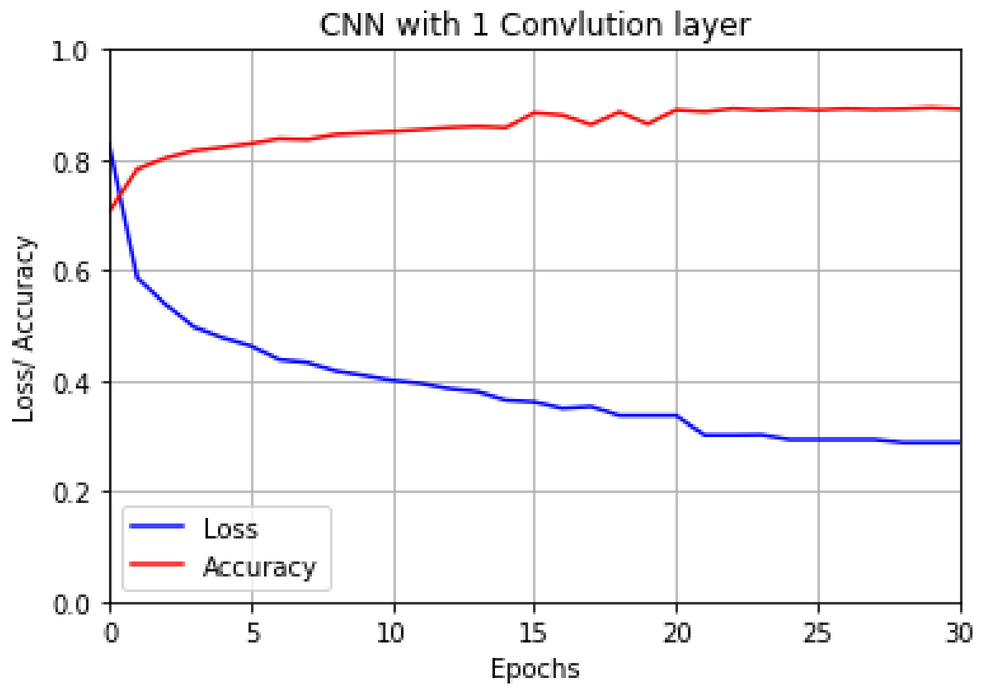
CNN with 1 Conv Layer

Hyper-parameters

Parameters	Value
Conv Layer	1
Pooling Layer	1
FC Layer	1
Activation	ReLU
Learning Rate	0.001/ 0.0001
Total Parameters	693,962

Evaluation Metrics

Data	Accuracy	Loss
Validation Set	0.89	0.28
Test Set	0.89	0.25



More Experiments: with Data Augmentation

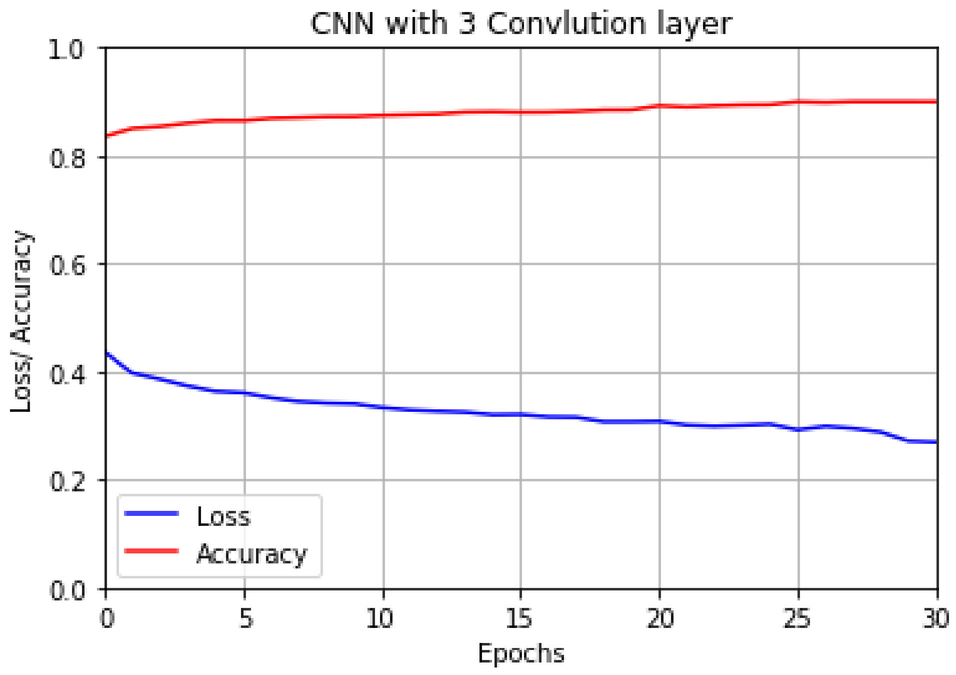
CNN with 3 Conv Layer

Hyper-parameters

Parameters	Value
Conv Layer	3
Pooling Layer	2
FC Layer	1
Activation	ReLU
Learning Rate	0.001/ 0.0001
Total Parameters	241,546

Evaluation Metrics

Data	Accuracy	Loss
Validation Set	0.91	0.26
Test Set	0.91	0.23



More Experiments: with Data Augmentation

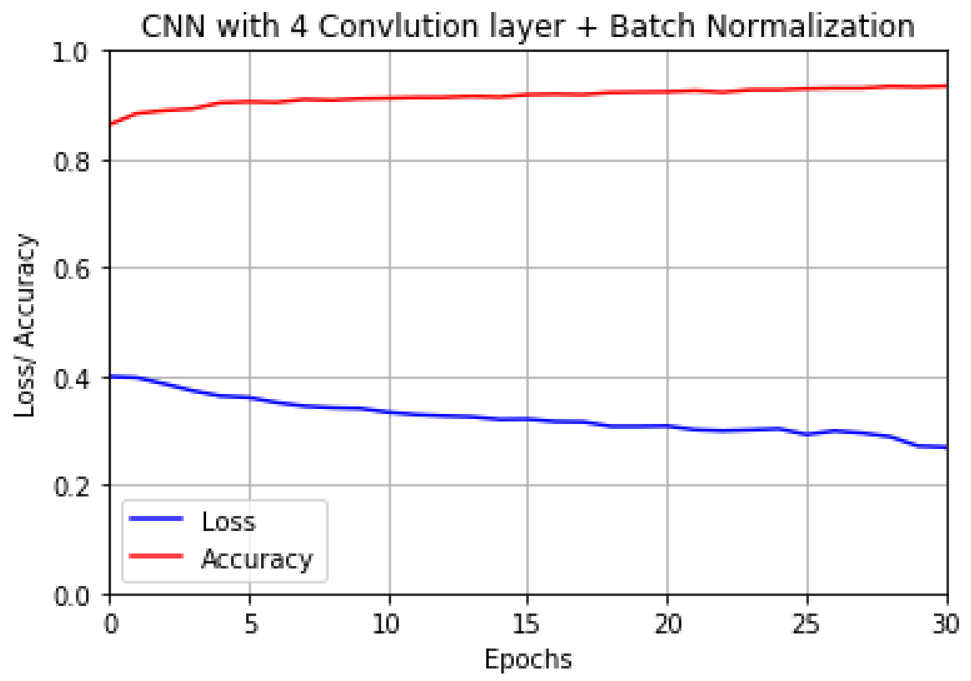
CNN with 4 Conv Layer + Batch Normalization

Hyper-parameters

Parameters	Value
Conv Layer	1
Pooling Layer	1
FC Layer	1
Activation	ReLU
Learning Rate	0.001/ 0.0001
Total Parameters	693,962

Evaluation Metrics

Data	Accuracy	Loss
Validation Set	0.93	0.18
Test Set	0.92	0.23



More Experiments: with Data Augmentation

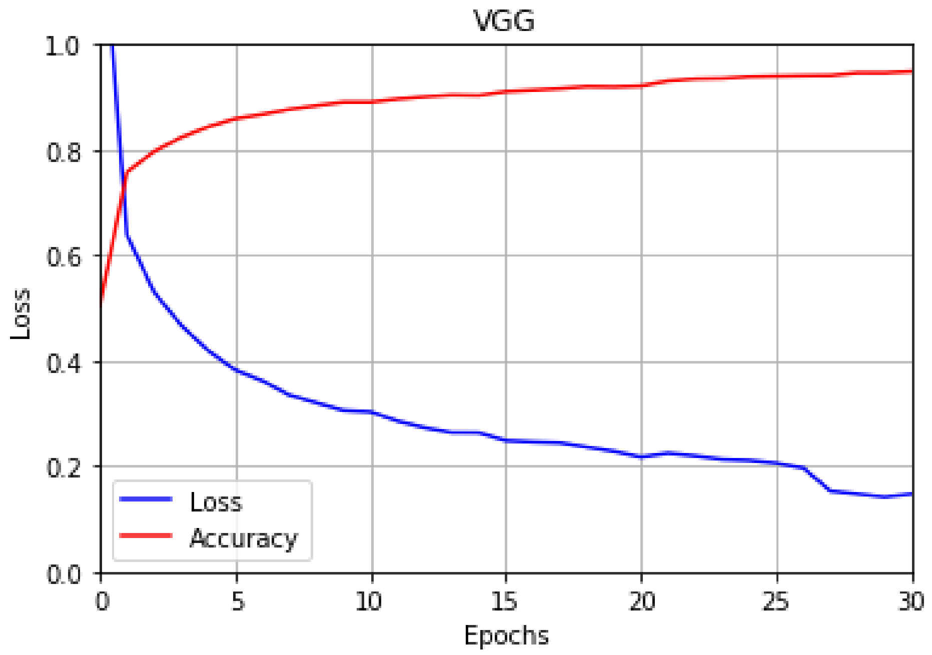
VGG like model

Hyper-parameters

Parameters	Value
Conv Layer	9
Pooling Layer	4
FC Layer	1
Activation	ReLU
Learning Rate	0.001/ 0.0001
Total Parameters	241,546

Evaluation Metrics

Data	Accuracy	Loss
Validation Set	0.94	0.14
Test Set	0.92	0.22



More Experiments: with Data Augmentation

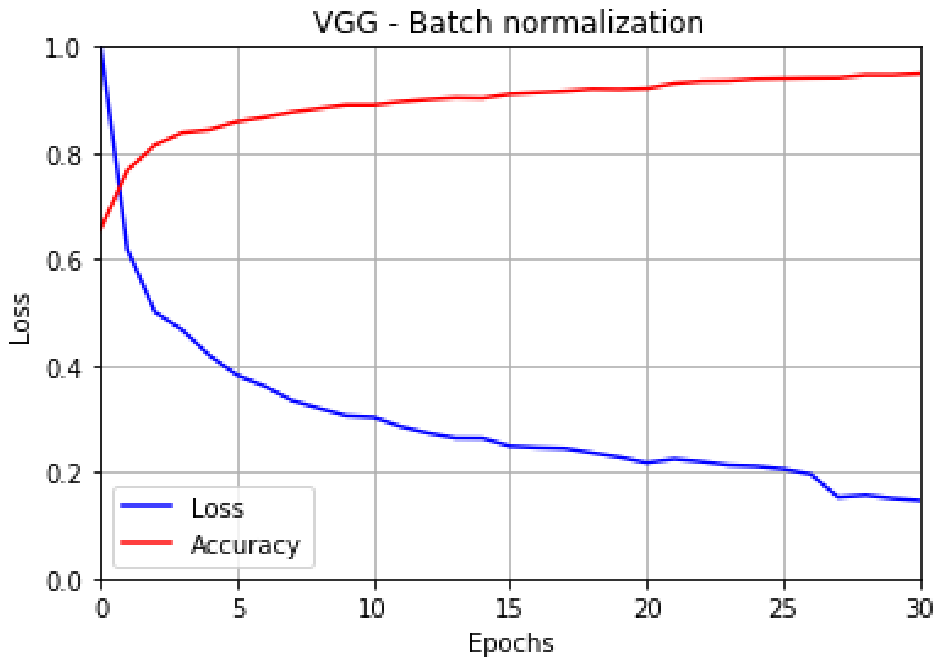
VGG + Batch normalization

Hyper-parameters

Parameters	Value
Conv Layer	9
Pooling Layer	4
FC Layer	1
Activation	ReLU
Learning Rate	0.001/ 0.0001
Total Parameters	693,962

Evaluation Metrics

Data	Accuracy	Loss
Validation Set	0.94	0.14
Test Set	0.93	0.18



More Experiments: with Data Augmentation

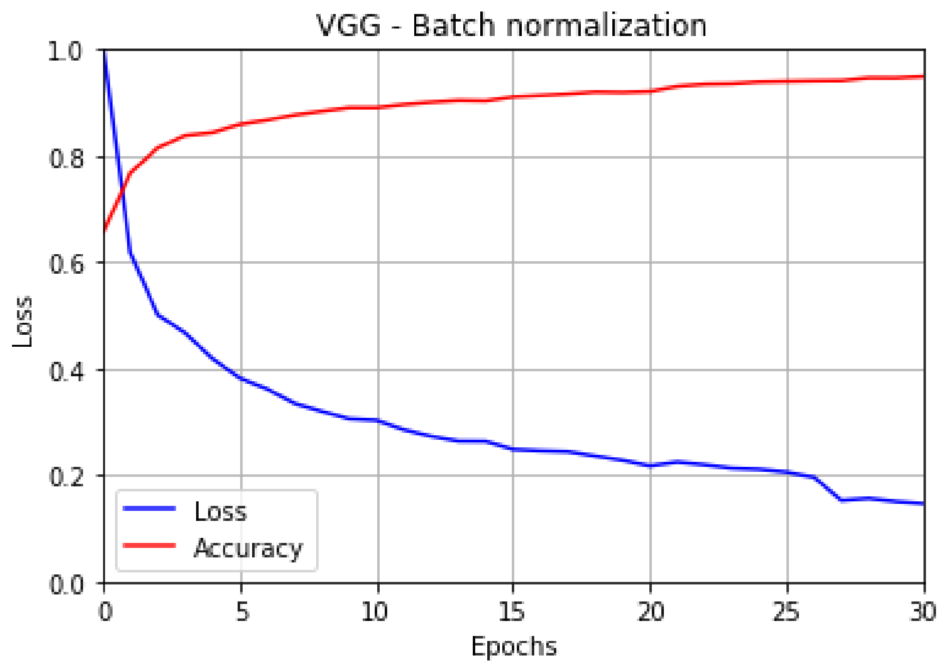
VGG + Batch normalization

Hyper-parameters

Parameters	Value
Conv Layer	9
Pooling Layer	4
FC Layer	1
Activation	ReLU
Learning Rate	0.001/ 0.0001
Total Parameters	693,962

Evaluation Metrics

Data	Accuracy	Loss
Validation Set	0.94	0.14
Test Set	0.93	0.18



More Experiments: with Data Augmentation

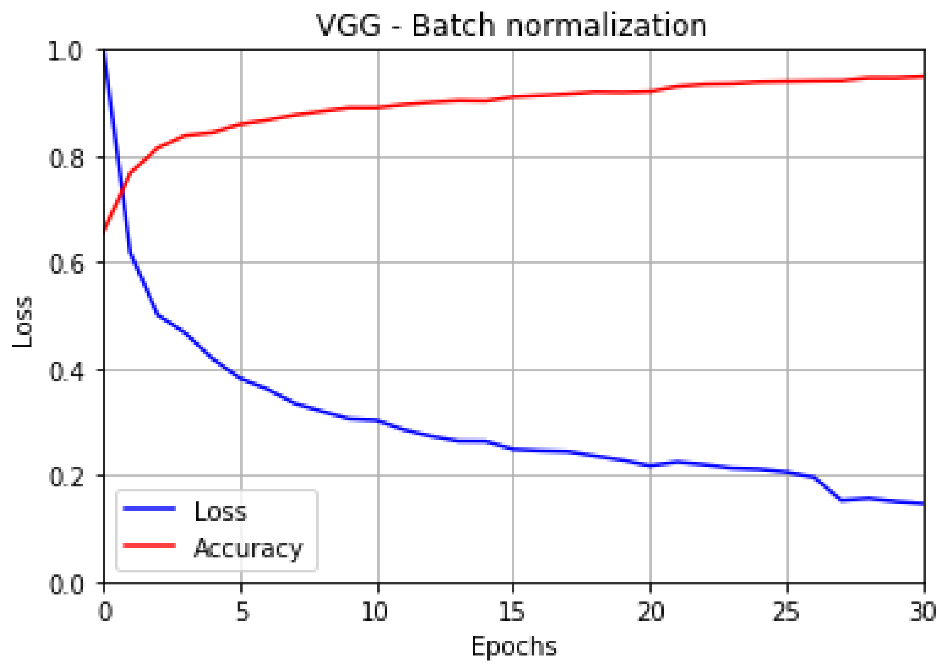
VGG + Batch normalization

Hyper-parameters

Parameters	Value
Conv Layer	9
Pooling Layer	4
FC Layer	1
Activation	ReLU
Learning Rate	0.001/ 0.0001
Total Parameters	693,962

Evaluation Metrics

Data	Accuracy	Loss
Validation Set	0.94	0.14
Test Set	0.93	0.18



Tools and Setting Up Infrastructure

1. Deep Learning Framework: TensorFlow and Keras
2. Machine Learning Framework: Scikit-learn
3. Python stack: NumPy, SciPy, Seaborn, Matplotlib
4. Programming: Python

My Workstation:

  - MacBook, Processor Intel i5
  - RAM 8G

Google Cloud Environment:

- Zone: us-west1-b
- Virtual Machine instance
- Virtual CPU: (n1-standard-1)
  - Intel i7-5930 3.5 GHz
  - System memory: 3.75 GB
- GPU: Nvidia Titan X

Observations: Accuracy and Loss

Model	Accuracy (%)	Loss
2 Layer Model	88	0.35
CNN - 1 Conv layer	91	0.25
CNN - 3 Conv layer	91	0.25
CNN - 4 Conv layer + BG	92	0.23
VGG	93	0.22
VGG + Batch Normalization	95	0.14

Learnings: Not taught in any deep learning course

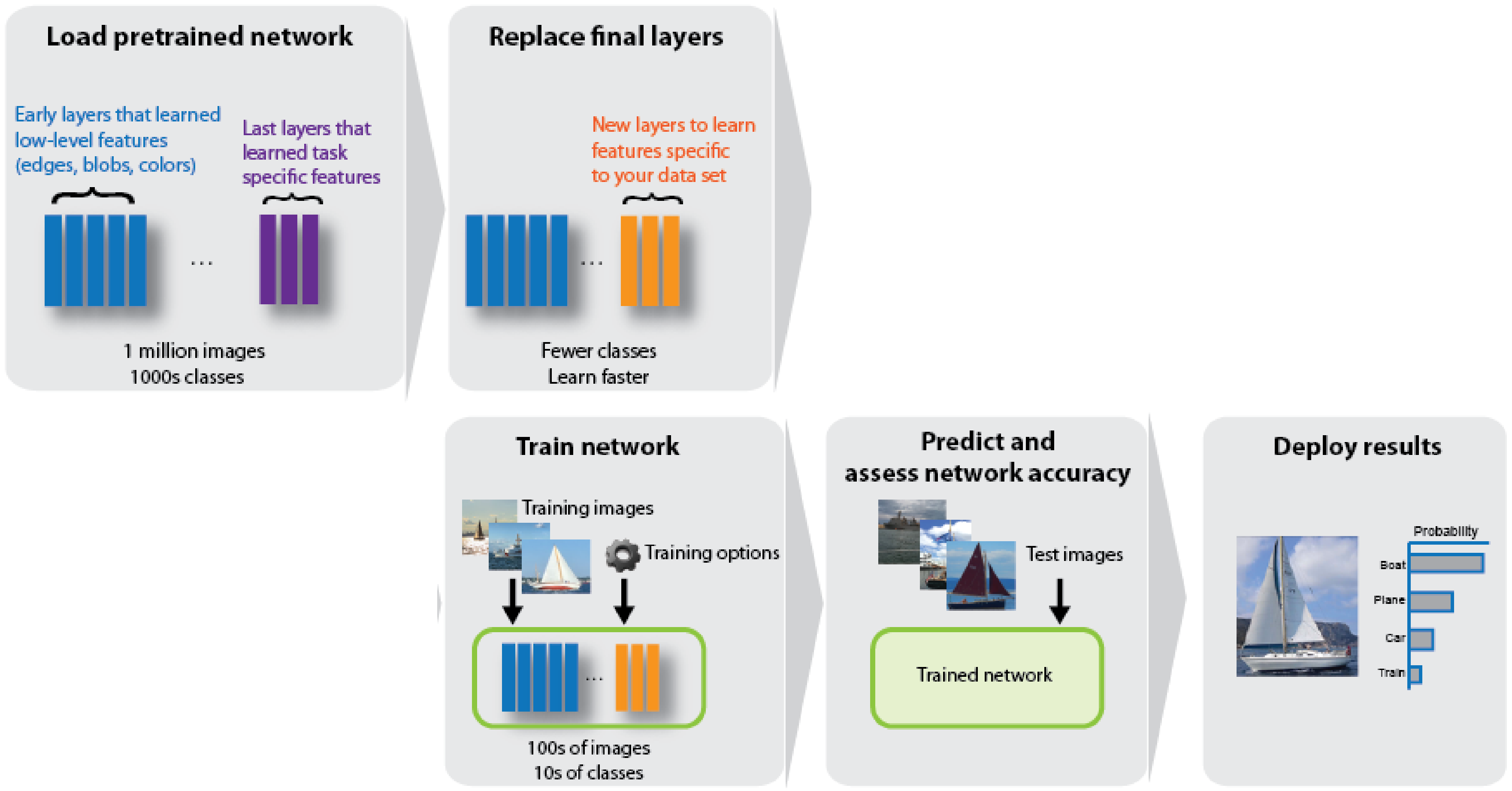
1. Understand data generating process, it gives you intuition about which model to choose and why.
2. Always do error analysis, it gives you clues how to modify your model to get rid of those errors.
3. The entire game of deep learning is about training and optimizing your hyperparameters.
4. Always try early stopping for regularization before getting into complex techniques like dropout.
5. Accuracy is not the best metric to evaluate a classifier. Try LogLoss, Confusion matrix, F1 score, AUC.
6. Deeper model learn faster than shallow models with less parameters (neurons).
7. Deeper models tend to generalize nicely on unseen data.

Future Work:

1. Transfer Learning
2. Hyperparameter Optimization
3. More deeper models with improved architectures.
    - Current trend towards getting rid of POOL/FC layers (just CONV layers)

Transfer Learning

- Use pre-trained layers in your models trained on big datasets like ImageNet Dataset (1 Million of images and 1,000 Classes)



References

- Alex Krizhevskv (2012). ImageNet Classification with Deep Convolutional Neural Networks.
- Karen Simonyan (2015). Very Deep Convolutional Networks for large-scale Image recognition.
- Kaming He. (2016). Deep Residual Learning for Image Recognition.
- Christian Szegedy. (2015). Going Deeper with Convolutions.
- Jason Wang. The Effectiveness of Data Augmentation in Image Classification using Deep Learning.
- Fei-Fei Li, Justin Johnson, Serena Yeung. Course: CS231n http://cs231n.stanford.edu
- Image Augmentation for Deep Learning with Keras. https://machinelearningmastery.com/image-augmentation-deep-learning-keras
- Fashion-MNIST Dataset. https://github.com/zalandoresearch/fashion-mnist
- Image Classification using Convolutional Neural Network. https://pradeepsinngh.github.io/documents/CS596_Report_Pradeep_Singh.pdf

That’s all folks!!! Thank you!