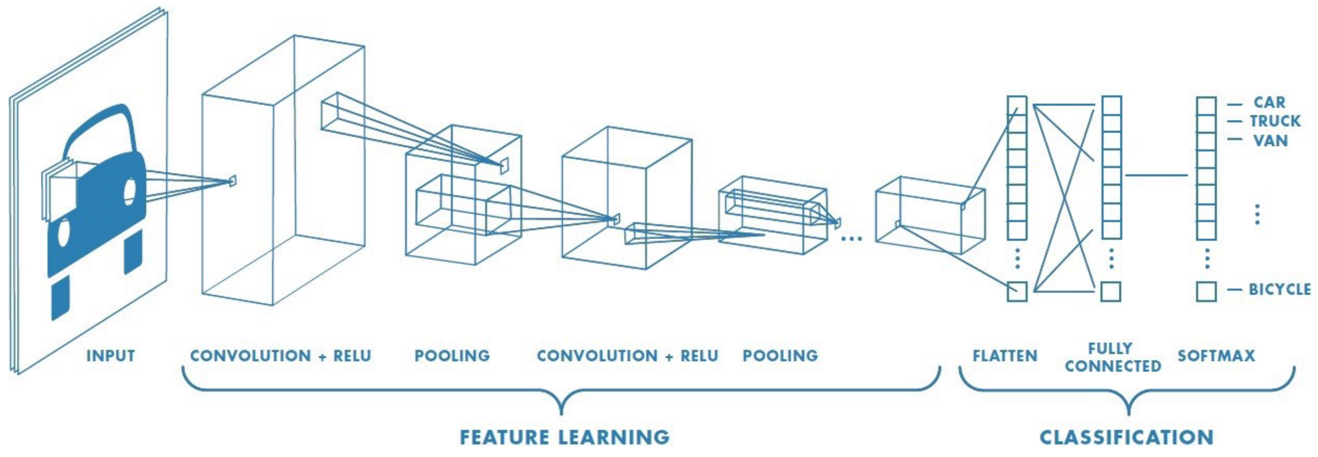
Report: Convolutional Neural Network

Olaoluwa Ogunleye[1*], Abd Al Rahman Al [1,2]

[1]Department of Mathematics, Clarkson University, Potsdam, NY

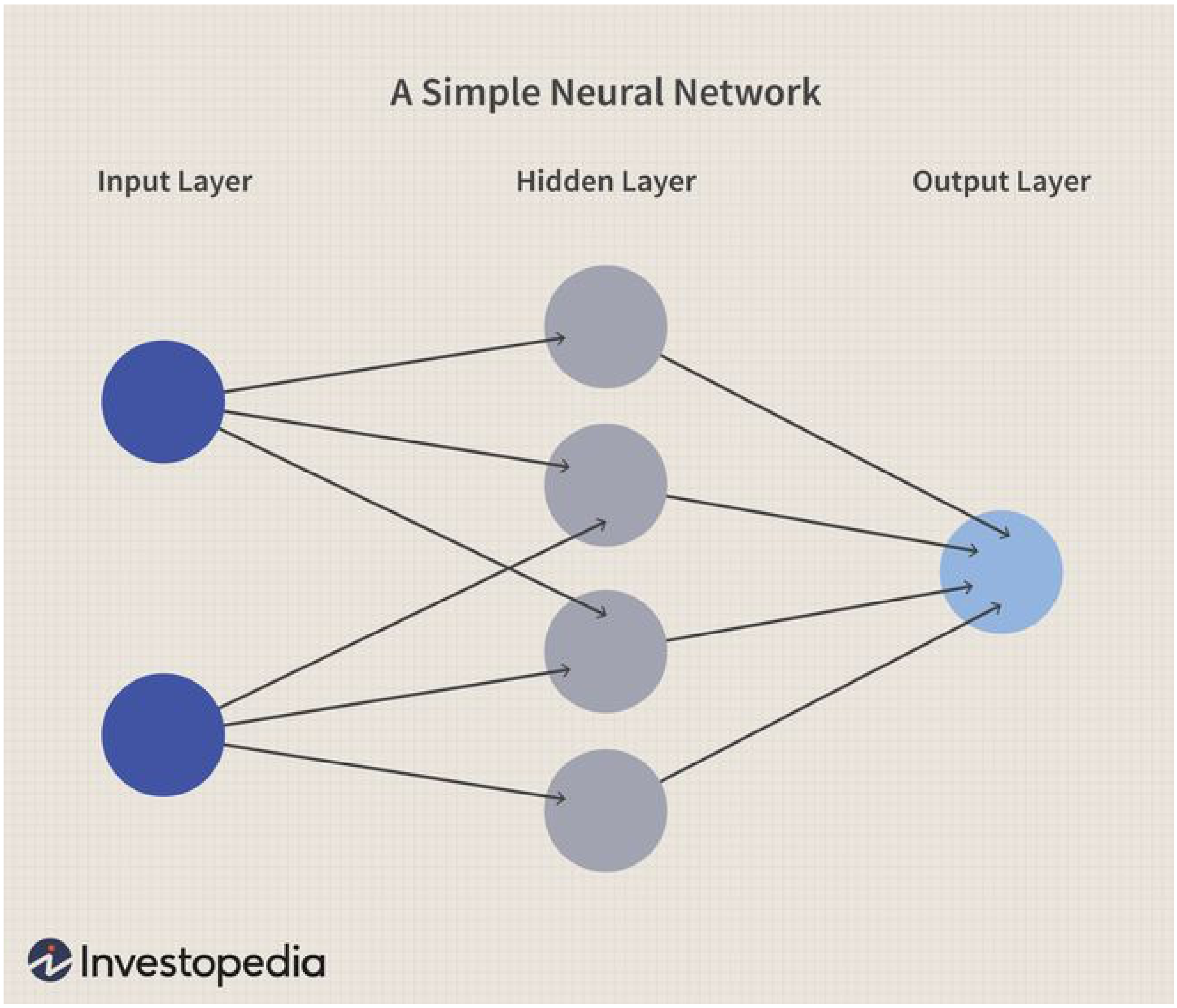
Introduction



What Is a Neural Network?

A neural network is a series of algorithms that tries to recognize underlying relationships the way a human brain does. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature.

Specifically, neural networks are able to adapt to dynamic input such that they generate optimal results without having to entirely reconstruct the output criteria. The concept of neural networks is gaining popularity both in academia and industry.



Neural networks are a series of algorithms that mimic the operations of an animal brain to recognize relationships between vast amounts of data. As such, they tend to resemble the connections of neurons and synapses found in the brain. They are used in a variety of applications in financial services, from forecasting and marketing research to fraud detection and risk assessment. Neural networks with several process layers are known as "deep" networks and are used for deep learning algorithms.

Basics of Neural Networks

A “neuron” in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis.

A neural network contains layers of interconnected nodes. Each node is known as a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

Multi-Layered Perceptron

In a multi-layered perceptron (MLP), perceptrons are arranged in interconnected layers. The input layer collects input patterns. The output layer has classifications or output signals to which input patterns may map

Hidden layers fine-tune the input weightings until the neural network's margin of error is minimal. It is hypothesized that hidden layers extrapolate salient features in the input data that have predictive power regarding the outputs. This describes feature extraction, which accomplishes a utility similar to statistical techniques such as principal component analysis.

Component of Neural Network

There are three main components: an input later, a processing layer, and an output layer. The inputs may be weighted based on various criteria. Within the processing layer, which is hidden from view, there are nodes and connections between these nodes, meant to be analogous to the neurons and synapses in an animal brain

Major types of neural networks

1. Convolutional Neural Network: A convolutional neural network is one adapted for analyzing and identifying visual data such as digital images or photographs
2. Recurrent Neural Network: A recurrent neural network is one adapted for analyzing time series data, event history, or temporal ordering.
3. Deep Neural Network: Also known as a deep learning network, a deep neural network, at its most basic, is one that involves two or more processing layers.

Computer Vision

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. In other words, it seeks to understand and automate tasks that the human visual system can do.

Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information that influences decision or actions such as the vision sensors of self-driving cars and delivery robots.

The goal is to enable machines to view the world as humans do, perceive it in a similar manner and even utilize the knowledge for a multitude of tasks such as Image & Video recognition, Image Analysis & Classification, Media Recreation, Recommendation Systems, Natural Language Processing, and more.

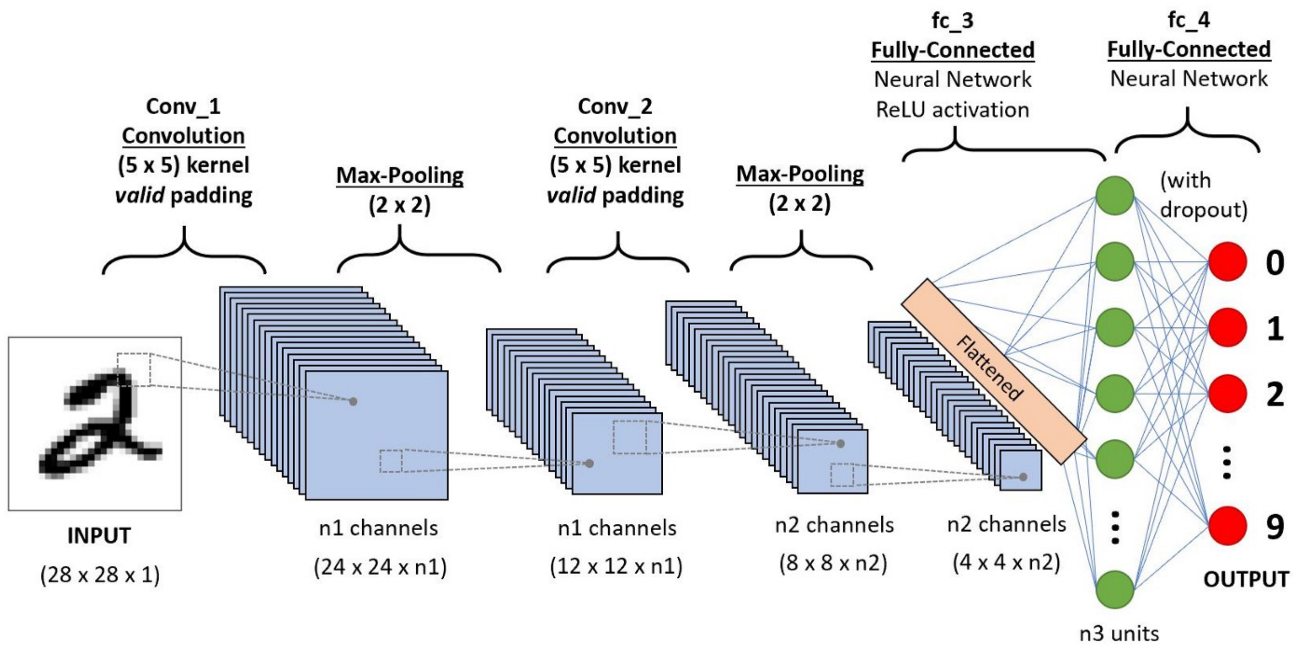
Convolutional Neural Network (ConvNet/CNN)

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of artificial neural network (ANN), usually applied to analyze visual imagery. CNNs are also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN), based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation-equivariant responses known as feature maps

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

A good thing about this is that the pre-processing required in a ConvNet is much lower as compared to other classification algorithms. This means that the network learns to optimize the filters (or kernels) through automated learning, whereas in traditional algorithms these filters are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage

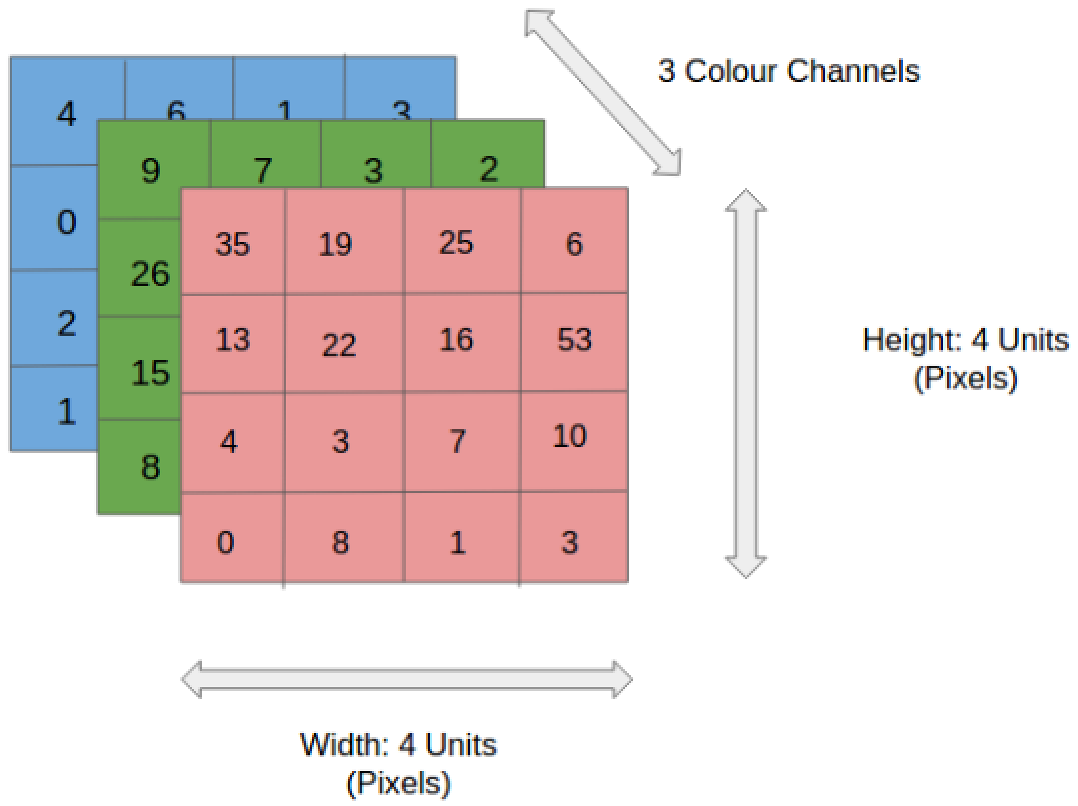
The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.



Understanding that an image is simply a matrix of pixel values, a ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of some filters.

The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

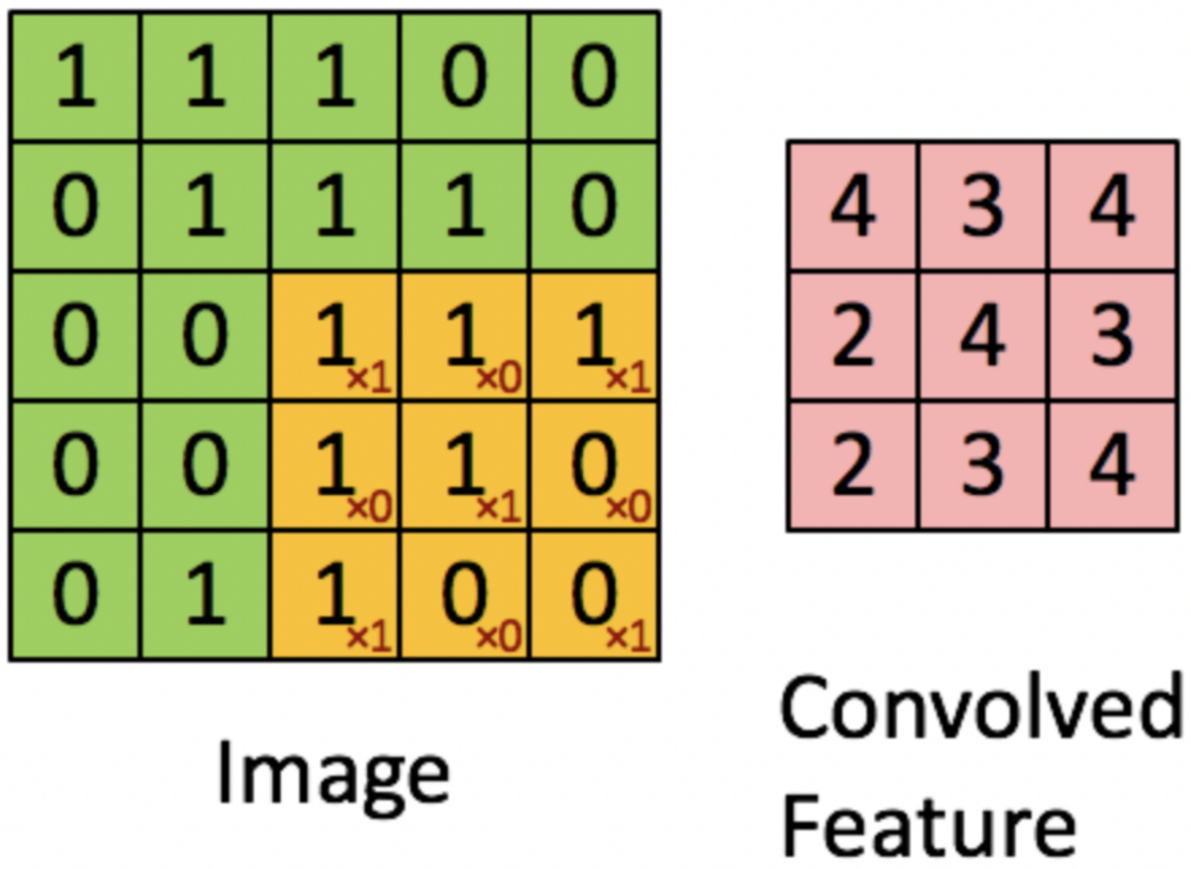
Input Image



An RGB image, like the one above. has three color planes - Red, Green, and Blue.

The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

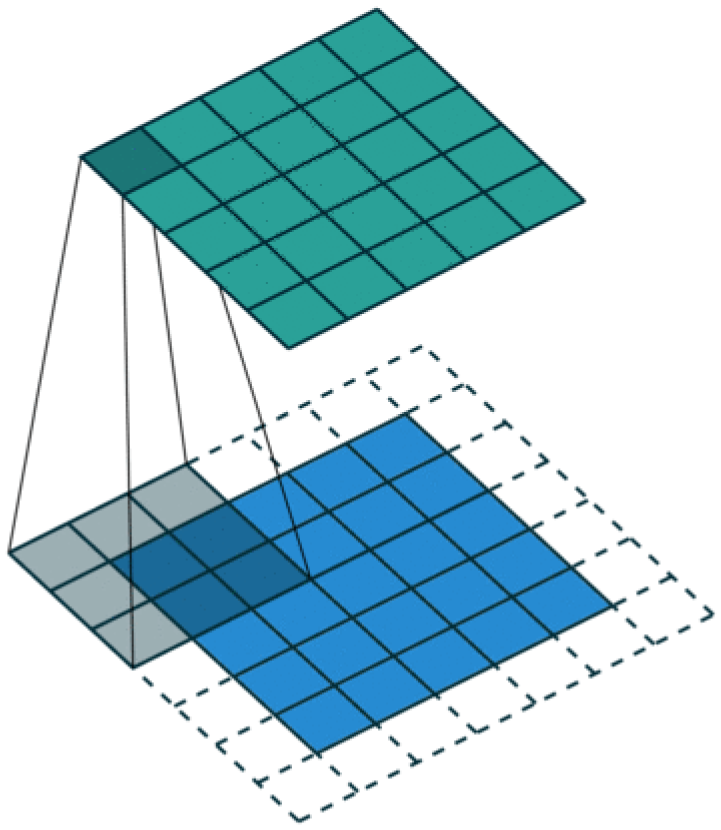
Convolution Layer - The Kernel



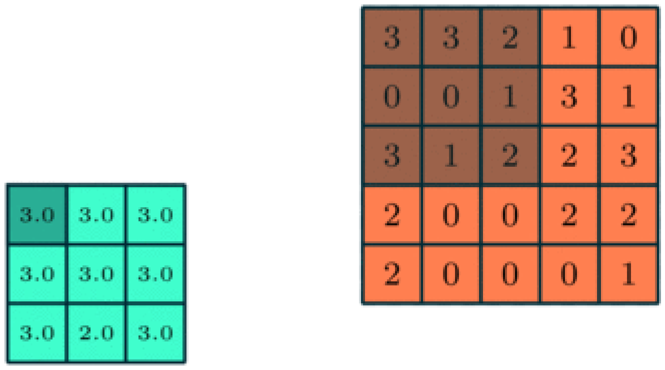
The above image shows convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature. The green section resembles our 5x5x1 input image, I. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the color yellow.

The Convolution Operation extracts high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset, similar to how we would.

There are two types of results to the operation - one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying *valid padding* in case of the former, or *same padding* in the case of the latter.



Pooling Layer



Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network.

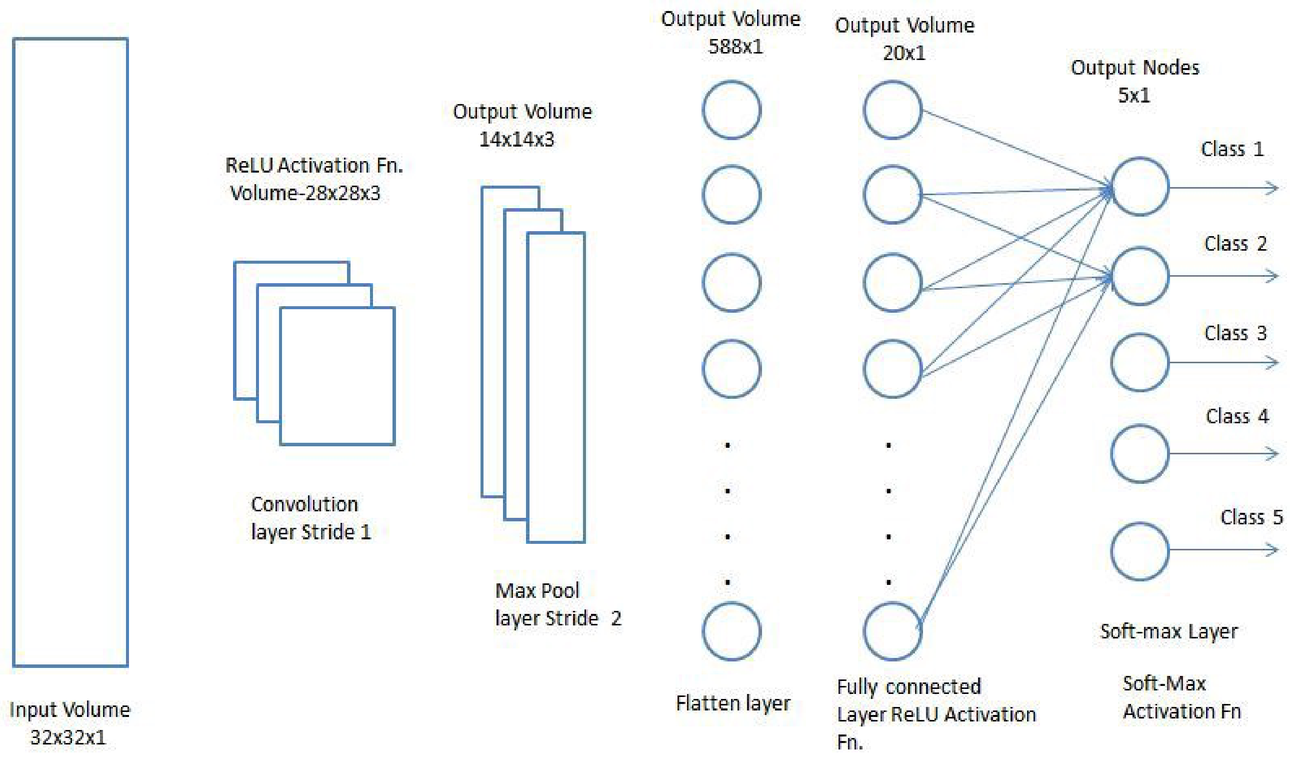
The pooling layer summarizes the features present in a region of the feature map generated by a convolution layer. So, further operations are performed on summarized features instead of precisely positioned features generated by the convolution layer. This makes the model more robust to variations in the position of the features in the input image

Types of Pooling Layers:

1. **Max Pooling**: Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.
2. **Average Pooling**: Average pooling computes the average of the elements present in the region of the feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch.
3. **Global Pooling**: Global pooling reduces each channel in the feature map to a single value. Thus, an nh x nw x nc feature map is reduced to 1 x 1 x nc feature map. This is equivalent to using a filter of dimensions nh x nw i.e. the dimensions of the feature map. Further, it can be either global max pooling or global average pooling.

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model.

Classification — Fully Connected Layer (FC Layer)



Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

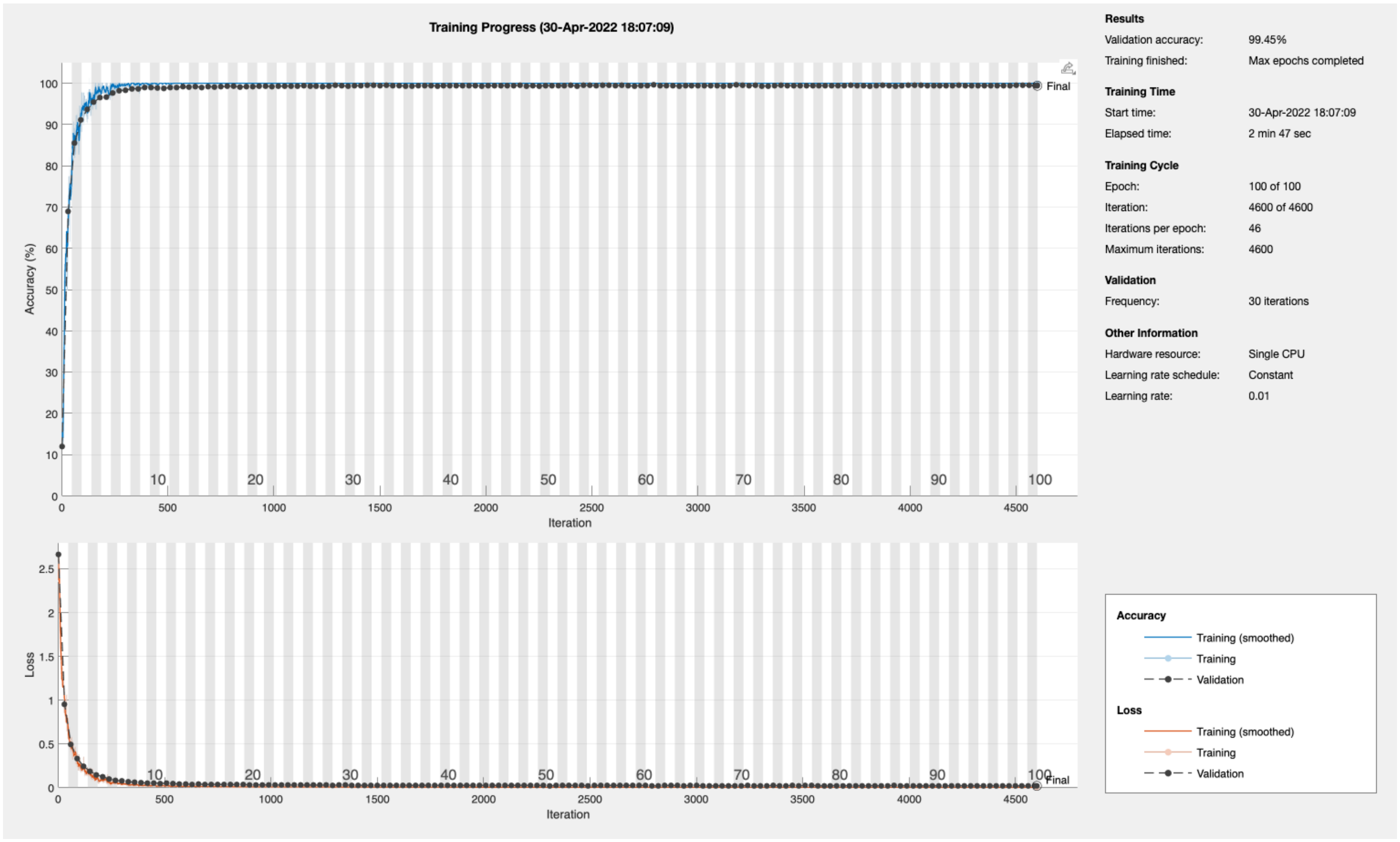
Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future. Some of them have been listed below:

1. **LeNet**: Recognizing simple digit images is the most classic application of LeNet as it was raised because of that.
2. **AlexNet**: AlexNet contained eight layers; the first five were convolutional layers, some of them followed by max-pooling layers, and the last three were fully connected layers. It used the non-saturating ReLU activation function, which showed improved training performance over tanh and sigmoid.
3. **GoogLeNet**: GoogLeNet is a convolutional neural network that is 22 layers deep.:

4. **ZFNet**: ZFNet is a classic convolutional neural network. The design was motivated by visualizing intermediate feature layers and the operation of the classifier. Compared to AlexNet, the filter sizes are reduced and the stride of the convolutions are reduced.

Matlab Example Implementation Result



Results

```
Validation accuracy:        99.45%
Training finished:          Max epochs completed
```

Training Time

```
Start time:                 30-Apr-2022 18:07:09
Elapsed time:               2 min 47 sec
```

Training Cycle

```
Epoch:                      100 of 100
Iteration:                  4600 of 4600
Iterations per epoch:       46
Maximum iterations:         4600
```

Validation

```
Frequency:                  30 iterations
```

Other Information

```
Hardware resource:          Single CPU
Learning rate schedule:     Constant
Learning rate:              0.01
```

```
close all, clear all
```

Step 1: Loading dataset

```
 digitDatasetPath = fullfile(matlabroot,'toolbox','nnet','nndemos', ...
     'nndatasets','DigitDataset');
 imds = imageDatastore(digitDatasetPath, ...
     'IncludeSubfolders',true,'LabelSource','foldernames');

%  Quick example
% figure; imshow(uint8(imds.Files{2005}))
```

Step 2: Splitting dataset into 60% Train, 20% Validation, 20% Test

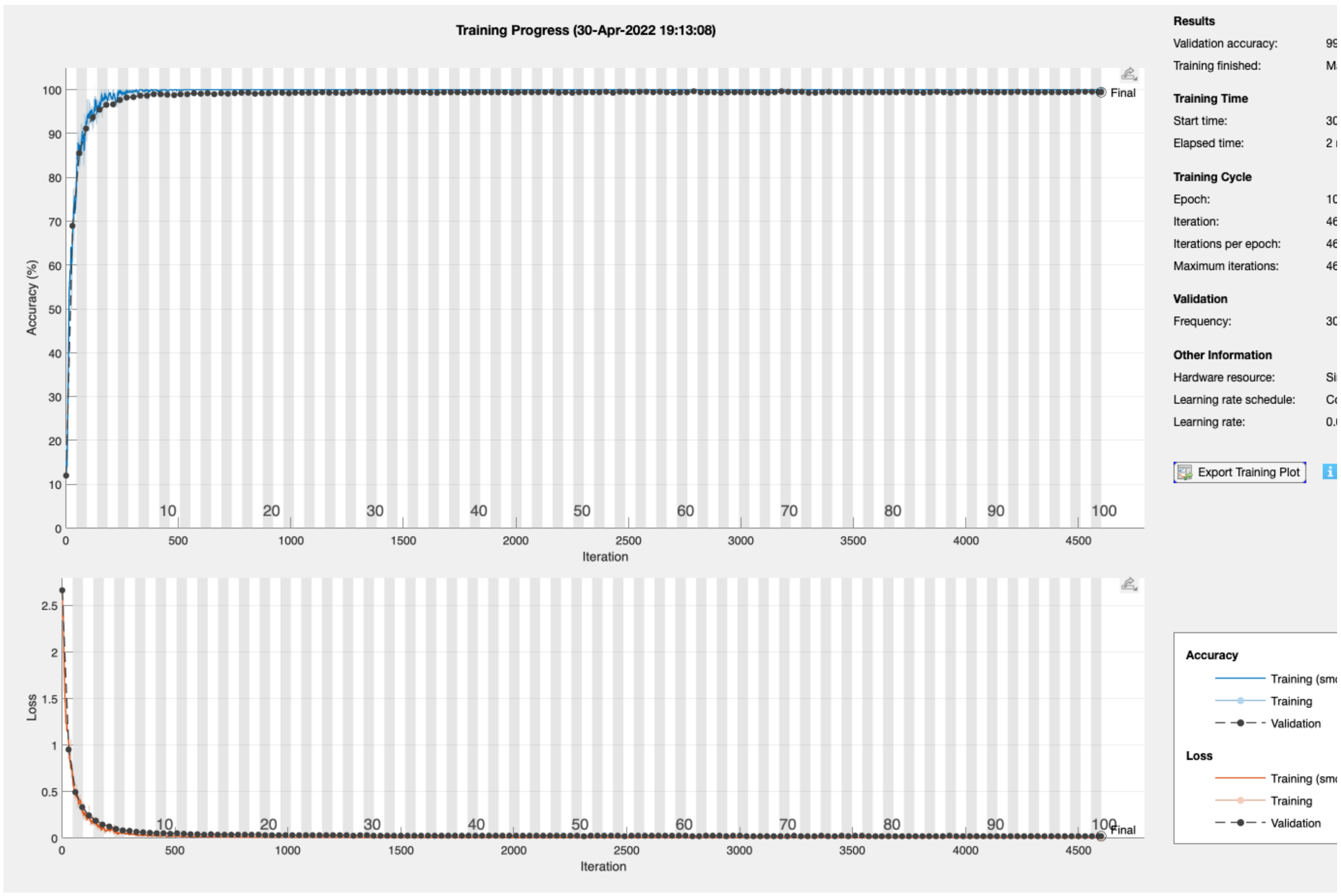
```
 fracTrainFiles = 0.6;
 fracValFiles = 0.2;
 fracTestFiles = 0.2;
 [imdsTrain, imdsValidation, imdsTest ] = splitEachLabel(imds, fracTrainFiles, fracValFiles, fracTestFiles, 'randomize');
```

Step 3: Create list of layers

```
 layers = [
     imageInputLayer([28 28 1])
     convolution2dLayer(3,10,'Padding','same')
     batchNormalizationLayer % This helps deal with variations in batch properties, doesn't matter much here.
     reluLayer % Relu layer choice is largely up to the user, pretty much any would work here.
     maxPooling2dLayer(2,'Stride',2)

     convolution2dLayer(3,10,'Padding','same')
     batchNormalizationLayer
     reluLayer

     fullyConnectedLayer(10)
     softmaxLayer % The results from the fully connected layer are converted to explicit probabilities by the soft max - it no
     classificationLayer];
```

Step 4: Train Model

```
 options = trainingOptions('sgdm', ...
     'InitialLearnRate',0.01, ...
     'MaxEpochs',100, ...
     'Shuffle','every-epoch', ...
     'ValidationData',imdsValidation, ...
     'ValidationFrequency',30, ...
     'Verbose',false, ...
     'Plots','training-progress');

 net = trainNetwork(imdsTrain,layers,options);
```



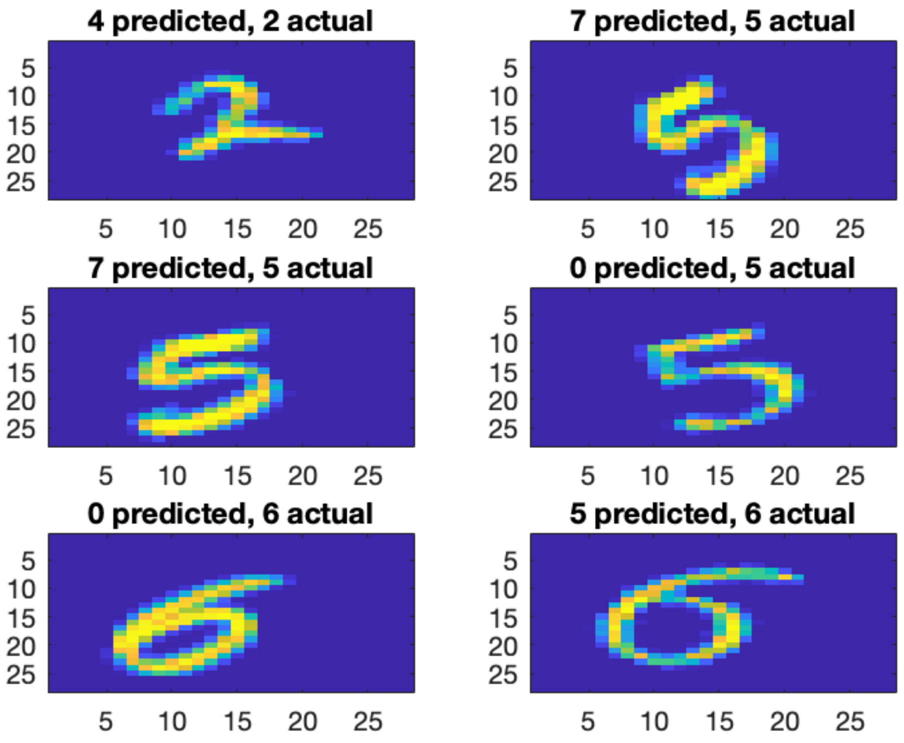
Results
Validation accuracy: 99
Training finished: M
Training Time
Start time: 30
Elapsed time: 2
Training Cycle
Epoch: 10
Iteration: 46
Iterations per epoch: 46
Maximum iterations: 46
Validation
Frequency: 30
Other Information
Hardware resource: Si
Learning rate schedule: Co
Learning rate: 0.0

Step 5: Test model

```
YPred = classify(net,imdsTest);
YTest = imdsTest.Labels;

accuracy = sum(YPred == YTest)/numel(YTest)
```

```
accuracy = 0.9965
```

```

ind = find(YPred ~= YTest);
figure;
for ii = 1:length(ind)
    subplot(3, floor(length(ind)/3),ii);
    imagesc(imdsValidation.readimage(ind(ii)));
    title([num2str(double(YPred(ind(ii)))-1), ' predicted, ', num2str(double(YTest(ind(ii)))-1), ' actual'])
end
```



```
Error using subplot
Index exceeds number of subplots.
```

References

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

https://www.investopedia.com/terms/n/neuralnetwork.asp

https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

https://en.wikipedia.org/wiki/AlexNet

https://www.mathworks.com/help/deeplearning/ref/googlenet.html

https://www.youtube.com/watch?v=lK9YyX-q32k