1. Explain convolutional neural network, and how does it work?

A CNN can have multiple layers, each of which learns to detect the different features of an input image. A filter or kernel is applied to each image to produce an output that gets progressively better and more detailed after each layer. In the lower layers, the filters can start as simple features.

network (CNN)

What is convolutional neural network (CNN or convnet)?

A convolutional neural network (CNN or convnet) is a subset of [machine learning](https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML). It is one of the various types of artificial [neural networks](https://www.techtarget.com/searchenterpriseai/definition/neural-network) which are used for different applications and data types. A CNN is a kind of network architecture for [deep learning](https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network) algorithms and is specifically used for [image recognition](https://www.techtarget.com/searchenterpriseai/definition/image-recognition) and tasks that involve the processing of [pixel](https://www.techtarget.com/whatis/definition/pixel) data.

There are other types of neural networks in deep learning, but for identifying and recognizing objects, CNNs are the network architecture of choice. This makes them highly suitable for computer vision ([CV](https://www.techtarget.com/searchenterpriseai/definition/machine-vision-computer-vision)) tasks and for applications where object recognition is vital, such as [self-driving cars](https://www.techtarget.com/searchenterpriseai/definition/driverless-car) and [facial recognition](https://www.techtarget.com/searchenterpriseai/definition/facial-recognition).

Inside convolutional neural networks

Artificial neural networks (ANNs) are a core element of deep learning algorithms. One type of an ANN is a recurrent neural network ([RNN](https://www.techtarget.com/searchenterpriseai/definition/recurrent-neural-networks)) that uses sequential or time series data as input. It is suitable for applications involving natural language processing ([NLP](https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP)), language translation, [speech recognition](https://www.techtarget.com/searchcustomerexperience/definition/speech-recognition) and image captioning.

The CNN is another type of neural network that can uncover key information in both time series and image data. For this reason, it is highly valuable for image-related tasks, such as image recognition, object classification and [pattern recognition](https://www.techtarget.com/whatis/definition/pattern-recognition). To identify patterns within an image, a CNN leverages principles from linear algebra, such as matrix multiplication. CNNs can also classify audio and [signal](https://www.techtarget.com/searchnetworking/definition/signal) data.

A CNN's architecture is analogous to the connectivity pattern of the human brain. Just like the brain consists of billions of neurons, CNNs also have neurons arranged in a specific way. In fact, a CNN's neurons are arranged like the brain's frontal lobe, the area responsible for processing visual stimuli. This arrangement ensures that the entire visual field is covered, thus avoiding the piecemeal image processing problem of traditional neural networks, which must be fed images in reduced-resolution pieces. Compared to the older networks, a CNN delivers better performance with image inputs, and also with speech or audio signal inputs.

Convolutional neural network, a subset of machine learning, is a type of artificial neural network.

CNN layers

A deep learning CNN consists of three layers: a convolutional layer, a pooling layer and a fully connected (FC) layer. The convolutional layer is the first layer while the FC layer is the last.

From the convolutional layer to the FC layer, the complexity of the CNN increases. It is this increasing complexity that allows the CNN to successively identify larger portions and more complex features of an image until it finally identifies the object in its entirety.

Convolutional layer. The majority of computations happen in the convolutional layer, which is the core building block of a CNN. A second convolutional layer can follow the initial convolutional layer. The process of convolution involves a [kernel](https://www.techtarget.com/searchdatacenter/definition/kernel) or filter inside this layer moving across the receptive fields of the image, checking if a feature is present in the image.

Over multiple iterations, the kernel sweeps over the entire image. After each iteration a [dot product](https://www.techtarget.com/whatis/definition/dot-product-scalar-product) is calculated between the input pixels and the filter. The final output from the series of dots is known as a feature map or convolved feature. Ultimately, the image is converted into numerical values in this layer, which allows the CNN to interpret the image and extract relevant patterns from it.

Pooling layer. Like the convolutional layer, the pooling layer also sweeps a kernel or filter across the input image. But unlike the convolutional layer, the pooling layer reduces the number of parameters in the input and also results in some information loss. On the positive side, this layer reduces complexity and improves the efficiency of the CNN.

Fully connected layer. The FC layer is where image classification happens in the CNN based on the features extracted in the previous layers. Here, fully connected means that all the inputs or nodes from one layer are connected to every activation unit or node of the next layer.

All the layers in the CNN are not fully connected because it would result in an unnecessarily dense network. It also would increase losses and affect the output quality, and it would be computationally expensive.

How convolutional neural networks work

A CNN can have multiple layers, each of which learns to detect the different features of an input image. A [filter](https://www.techtarget.com/whatis/definition/filter) or kernel is applied to each image to produce an output that gets progressively better and more detailed after each layer. In the lower layers, the filters can start as simple features.

At each successive layer, the filters increase in complexity to check and identify features that uniquely represent the input object. Thus, the output of each convolved image -- the partially recognized image after each layer -- becomes the input for the next layer. In the last layer, which is an FC layer, the CNN recognizes the image or the object it represents.

With convolution, the input image goes through a set of these filters. As each filter activates certain features from the image, it does its work and passes on its output to the filter in the next layer. Each layer learns to identify different features and the operations end up being repeated for dozens, hundreds or even thousands of layers. Finally, all the image data progressing through the CNN's multiple layers allow the CNN to identify the entire object.

A table depicting the differences between a convolutional neural network vs. recurrent neural network.

CNNs vs. neural networks

The biggest problem with regular neural networks (NNs) is a lack of scalability. For smaller images with fewer color channels, a regular NN may produce satisfactory results. But as the size and complexity of an image increases, the need for computational power and resources also increases which necessitates a [larger and more expensive NN](https://www.techtarget.com/searchenterpriseai/feature/Limitations-of-neural-networks-grow-clearer-in-business).

Moreover, the problem of overfitting also arises over time, wherein the NN tries to learn too many details in the training data. It may also end up learning the noise in the data, which affects its performance on test data sets. Ultimately, the NN fails to identify the features or patterns in the data set and thus the object itself.

In contrast, a CNN uses parameter sharing. In each layer of the CNN, each node connects to another. A CNN also has an associated weight; as the layers' filters move across the image, the weights remain fixed -- a condition known as parameter sharing. This makes the whole CNN system less computationally intensive than an NN system.

Benefits of using CNNs for deep learning

Deep learning is a subset of [machine learning](https://www.techtarget.com/searchenterpriseai/post/Introduction-to-using-machine-learning) that uses neural networks with at least three layers. Compared to a network with just one layer, a network with multiple layers can deliver more accurate results. Both RNNs and CNNs are used in deep learning, depending on the application.

For image recognition, image classification and [computer vision (CV) applications](https://www.techtarget.com/searchenterpriseai/ehandbook/Computer-vision-AI-looks-beyond-the-narrow-into-the-mainstream), CNNs are particularly useful because they provide highly accurate results, especially when a lot of data is involved. The CNN also learns the object's features in successive iterations as the object data moves through the CNN's many layers. This direct (and deep) learning eliminates the need for manual feature extraction ([feature engineering](https://www.techtarget.com/searchdatamanagement/definition/feature-engineering)).

CNNs can be retrained for new recognition tasks and built on preexisting networks. These advantages open up new opportunities to use CNNs for real-world applications without increasing computational complexities or costs.

2. How does refactoring parts of your neural network definition favor you?

A DNN refactoring defines (a) the transformation of the DNN's architecture, i.e., the number and size of its layers, and (b) the distillation of the learned relationships between the input features and function outputs of the original to train the transformed network

Deep neural networks are a powerful category of machine learning algorithms implemented by stacking layers of neural networks along the depth and width of smaller architectures.

3. What does it mean to flatten? Is it necessary to include it in the MNIST CNN? What is the reason for this?

Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector. The flattened matrix is fed as input to the fully connected layer to classify the image.

A flatten layer collapses the spatial dimensions of the input into the channel dimension. For example, if the input to the layer is an H-by-W-by-C-by-N-by-S array (sequences of images), then the flattened output is an (H\*W\*C)-by-N-by-S array.

Convolutional Neural Networks have changed the way we classify images. It is being used in almost all the computer vision tasks. From 2012, CNN’s have ruled the Imagenet competition, dropping the classification error rate each year. MNIST is the most studied dataset ([link](https://www.kaggle.com/benhamner/d/benhamner/nips-papers/popular-datasets-over-time)).

The state of the art result for MNIST dataset has an accuracy of 99.79%. In this article, we will achieve an accuracy of 99.55%.

What is the MNIST dataset?

MNIST dataset contains images of handwritten digits. It has 60,000 grayscale images under the training set and 10,000 grayscale images under the test set. We will use the Keras library with Tensorflow backend to classify the images.

What is a Convolutional Neural Network?

A convolution in CNN is nothing but a element wise multiplication i.e. dot product of the image matrix and the filter.

In the above example, the image is a 5 x 5 matrix and the filter going over it is a 3 x 3 matrix. A convolution operation takes place between the image and the filter and the convolved feature is generated. Each filter in a CNN, learns different characteristic of an image.

Installing Keras

Keras is a high-level neural network API, written in Python which runs on top of either Tensorflow or Theano. You can install Keras from [here](https://keras.io/#installation).

Tensorflow was developed by the Google Brain team. To learn more about it, visit there official [website](https://www.tensorflow.org/).

Keras was written to simplify the construction of neural nets, as tensorflow’s API is very verbose. Keras makes everything very easy and you will see it in action below. If you want to explore the tensorflow implementation of the MNIST dataset, you can find it [here](https://www.tensorflow.org/get_started/mnist/pros).

Implementation

First, we import all the necessary libraries required.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.optimizers import Adam

from keras.layers.normalization import BatchNormalization

from keras.utils import np\_utils

from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D, GlobalAveragePooling2D

from keras.layers.advanced\_activations import LeakyReLU

from keras.preprocessing.image import ImageDataGenerator

The MNIST dataset is provided by Keras.

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

The shape of X\_train is (60000, 28, 28). Each image has 28 x 28 resolution. The shape of X\_test is (10000, 28, 28).

The input shape that a CNN accepts should be in a specific format. If you are using Tensorflow, the format should be (batch, height, width, channels). If you are using Theano, the format should be (batch, channels, height, width).

So, let’s reshape our input.

X\_train = X\_train.reshape(X\_train.shape[0], 28, 28, 1)

X\_test = X\_test.reshape(X\_test.shape[0], 28, 28, 1)

X\_train = X\_train.astype('float32')

X\_test = X\_test.astype('float32')

X\_train/=255

X\_test/=255

Now the shape of X\_train is (60000, 28, 28, 1). As all the images are in grayscale, the number of channels is 1. If it was a color image, then the number of channels would be 3 (R, G, B).

Here we’ve rescaled the image data so that each pixel lies in the interval [0, 1] instead of [0, 255]. It is always a good idea to normalize the input so that each dimension has approximately the same scale.

Now, we need to one-hot encode the labels i.e. Y\_train and Y\_test. In one-hot encoding an integer is converted to an array which contains only one ‘1’ and the rest elements are ‘0’.

number\_of\_classes = 10

Y\_train = np\_utils.to\_categorical(y\_train, number\_of\_classes)

Y\_test = np\_utils.to\_categorical(y\_test, number\_of\_classes)

Y\_train[0] = [0., 0., 0., 0., 0., 1., 0., 0., 0., 0.] since the label representated by it is 5.

Let’s create the model that will classify the images (the most interesting part!!).

# Three steps to create a CNN

# 1. Convolution

# 2. Activation

# 3. Pooling

# Repeat Steps 1,2,3 for adding more hidden layers

# 4. After that make a fully connected network

# This fully connected network gives ability to the CNN

# to classify the samples

model = Sequential()

model.add(Conv2D(32, (3, 3), input\_shape=(28,28,1)))

model.add(BatchNormalization(axis=-1))

model.add(Activation('relu'))

model.add(Conv2D(32, (3, 3)))

model.add(BatchNormalization(axis=-1))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Conv2D(64,(3, 3)))

model.add(BatchNormalization(axis=-1))

model.add(Activation('relu'))

model.add(Conv2D(64, (3, 3)))

model.add(BatchNormalization(axis=-1))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Flatten())

# Fully connected layer

model.add(Dense(512))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Dropout(0.2))

model.add(Dense(10))

model.add(Activation('softmax'))

Keras allows us to specify the number of filters we want and the size of the filters. So, in our first layer, 32 is number of filters and (3, 3) is the size of the filter. We also need to specify the shape of the input which is (28, 28, 1), but we have to specify it only once.

The second layer is the Activation layer. We have used ReLU (rectified linear unit) as our activation function. ReLU function is f(x) = max(0, x), where x is the input. It sets all negative values in the matrix ‘x’ to 0 and keeps all the other values constant. It is the most used activation function since it reduces training time and prevents the problem of vanishing gradients.

The third layer is the MaxPooling layer. MaxPooling layer is used to down-sample the input to enable the model to make assumptions about the features so as to reduce over-fitting. It also reduces the number of parameters to learn, reducing the training time.

It’s a best practice to always do BatchNormalization. BatchNormalization normalizes the matrix after it is been through a convolution layer so that the scale of each dimension remains the same. It reduces the training time significantly.

After creating all the convolutional layers, we need to flatten them, so that they can act as an input to the Dense layers.

Dense layers are keras’s alias for Fully connected layers. These layers give the ability to classify the features learned by the CNN.

Dropout is the method used to reduce overfitting. It forces the model to learn multiple independent representations of the same data by randomly disabling neurons in the learning phase. In our model, dropout will randomnly disable 20% of the neurons.

The second last layer is the Dense layer with 10 neurons. The neurons in this layer should be equal to the number of classes we want to predict as this is the output layer.

The last layer is the Softmax Activation layer. Softmax activation enables us to calculate the output based on the probabilities. Each class is assigned a probability and the class with the maximum probability is the model’s output for the input.

4. What exactly does NCHW stand for?

NCHW stands for: batch N, channels C, depth D, height H, width W. It is a way to store multidimensional arrays / data frames / matrix into memory, which can be considered as a 1-D array.

5. Why are there 7\*7\*(1168-16) multiplications in the MNIST CNN's third layer?

There are 1168 parameters for that layer, and ignoring the 16 parameters (=number of filters) of the bias, the (1168-16) parameters is applied to the 7x7 grid.

6.Explain definition of receptive field?

The receptive field is the area of an image that is involved in the calculation of a layer.

7. What is the scale of an activation's receptive field after two stride-2 convolutions? What is the reason for this?

The size of the receptive field increases the deeper we are in the network. After two stride 2 convolutions, the receptive field is 7x7.

8. What is the tensor representation of a color image?

An image is made of many pixels, each of them with a particular color. Modern computers can show up to 16.7 millions of different colors. It is impossible to store each one of these colors personnally, they are instead represented as a combination of three primary colors: red, green and blue (RGB color model).

9. How does a color input interact with a convolution?

A convolutional "filter" consists of one or more kernels, and the convolution between an image and a filter can be computed by taking all of the rank-2 outputs of convolving the image with each kernel, and stacking up to create a rank-3 tensor (has height, width, and "depth", or multiple channels)