**REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE**
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider-BISKRA.

**Faculté des sciences exactes et des sciences de la nature et de la vie**

**Département d'informatique**

# Machine Learning (ML)

**Dr. TIBERMACINE Ahmed**

# Content Summary

# Chapter 3: Deep Learning (DL)

# Introduction

- In this chapter, we'll dive into the foundations of deep learning, placing a special spotlight on Convolutional Neural Networks (CNNs).

- We 'll get a grasp of what deep learning is all about, especially when it comes to dealing with images.

- By the end of this chapter, you'll be well-versed in how CNNs function and their crucial role in tasks like recognizing objects in pictures.

# Learning Objectives

- Gain insights into deep learning, its applications, and the emergence of CNNs.
- Comprehend the architecture and training of CNNs for image-related tasks.

# Deep Learning (DL)

- **What is Deep Learning?**

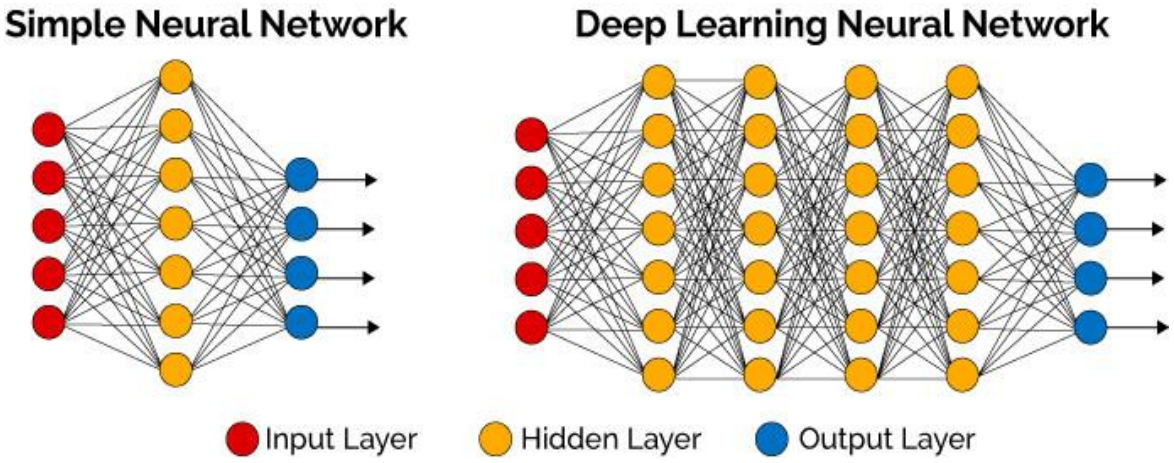
- Deep Learning is a subfield of machine learning and artificial intelligence that revolves around training **artificial neural networks**, particularly deep neural networks, to perform complex tasks.

- These deep neural networks consist of multiple layers, which enable them to capture intricate patterns in data.

# Deep Learning (DL)

## Deep Neural Networks (DNN)

- Deep neural networks are characterized by having many hidden layers.
- These layers introduce a hierarchical representation of data.
- A deep neural network consists of multiple layers:




- Mathematically, a deep neural network can be represented as a composition of functions:
  - $F(x)=f_n(f_{n-1}(\ldots f_2(f_1(x;W_1,b_1);W_2,b_2)\ldots;W_{n-1},b_{n-1});W_n,b_n)$

# Deep Learning (DL)

- **Panorama of Prominent Deep Learning Models**
  - Convolutional Neural Networks (CNNs)
  - Long Short-Term Memory Networks (LSTMs)
  - Autoencoders
  - Generative Adversarial Networks (GANs)
  - Transformers
  - Deep Reinforcement Learning (DRL)
  - Capsule Networks (CapsNets)

# Deep Learning (DL)

- **Differences between DNN and ML**
- **Scope:**
  - **ML**: is a broader field that encompasses various techniques for teaching a computer system to learn from data and make predictions.

    It includes both traditional statistical methods and modern techniques like deep learning.

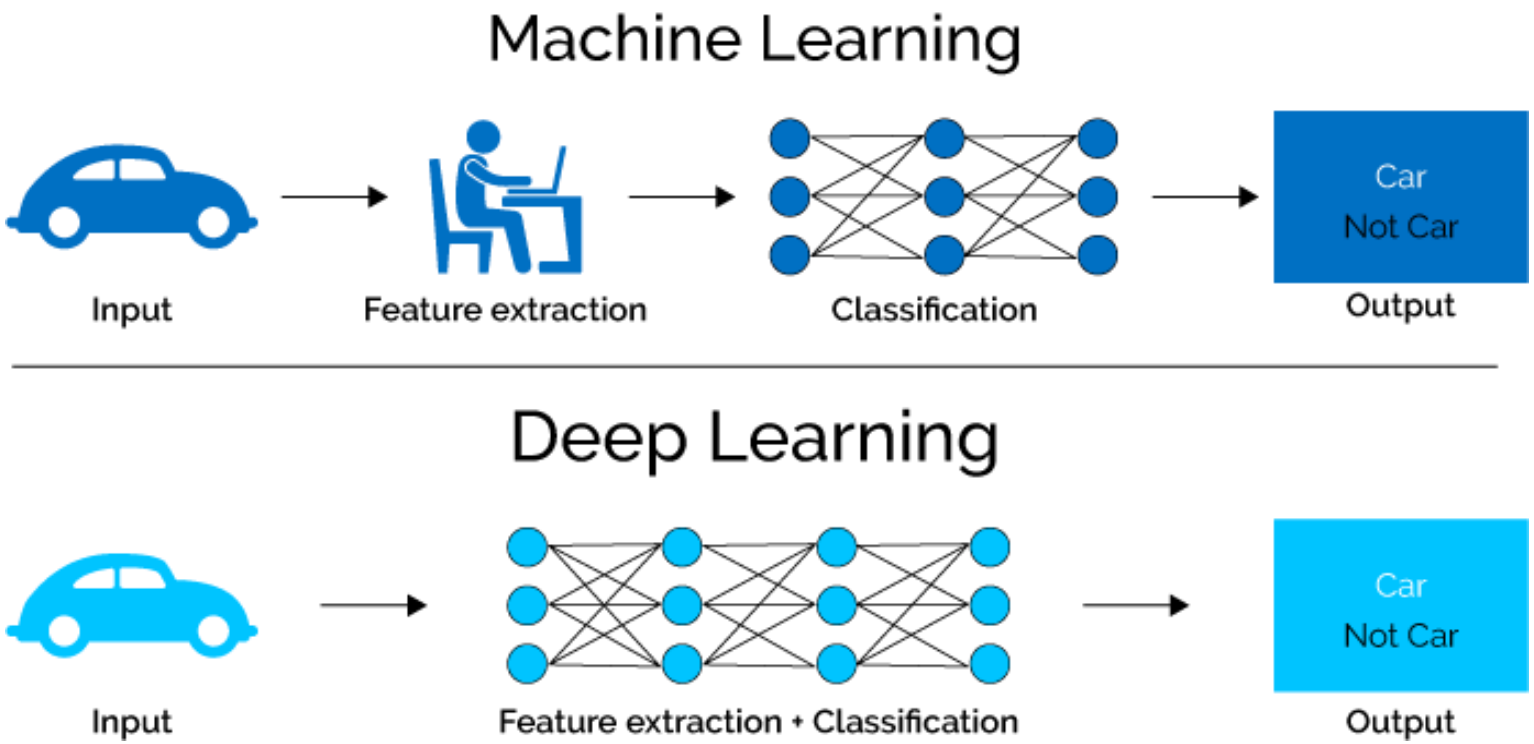
  - **DNN**: are a specific subset of machine learning techniques that are based on artificial neural networks with multiple hidden layers.

# Deep Learning (DL)

- **Differences between DNN and ML**
- **Model Complexity:**
  - **ML:** ML models can be relatively simple.
  - **DNN**: DNNs are characterized by their deep architectures, typically consisting of many layers.

# Deep Learning (DL)

- **Differences between DNN and ML**
- **Feature Engineering:**



# Deep Learning (DL)

- **Differences between DNN and ML**

- **Data Requirements:**
  - **ML**: ML models can work with smaller datasets effectively, as they often rely on hand-crafted features.
  - **DNN**: DNNs tend to require larger datasets for training, as they need substantial amounts of data to learn complex patterns.

# Deep Learning (DL)

- **Differences between DNN and ML**
- **Training Process:**
  - **ML**: Training ML models is often quicker and less computationally intensive compared to training deep neural networks.
  - **DNNs:** Training DNNs can be computationally expensive and time-consuming, often requiring specialized hardware like GPUs or TPUs.

# Deep Learning (DL)

- **Differences between DNN and ML**
- **Interpretability:**

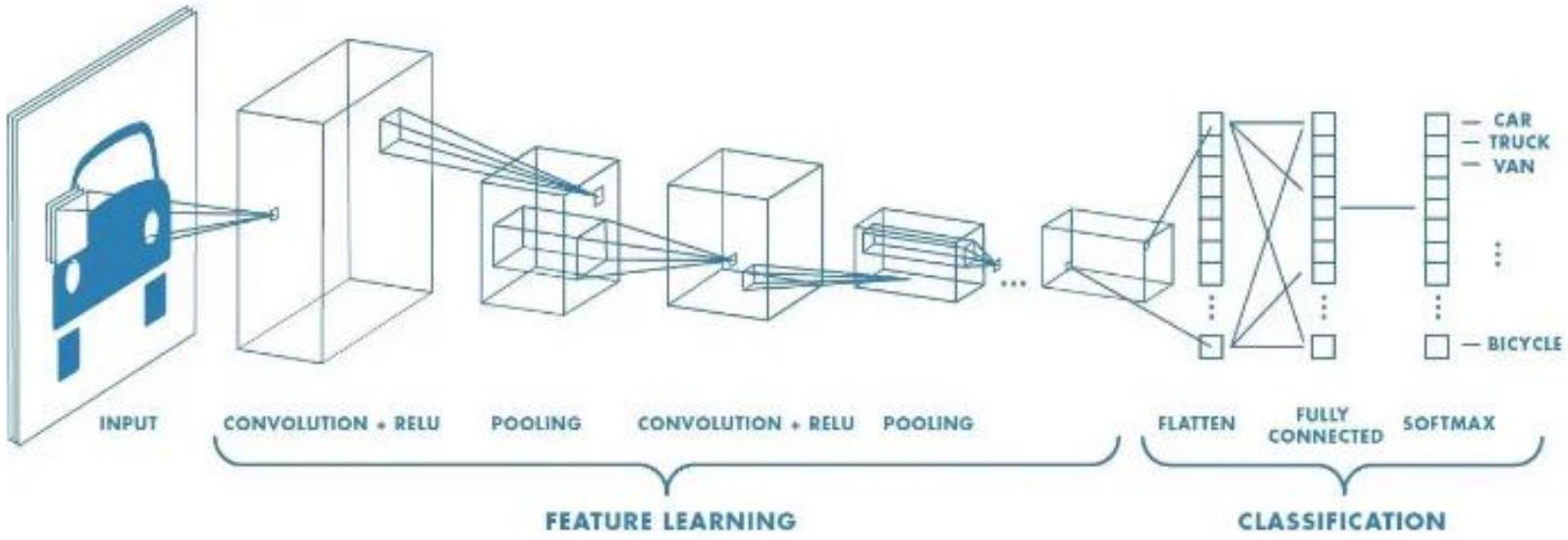
  - **ML:** ML models are relatively interpretable, meaning it's easier to understand the factors influencing their predictions.
  - **DNNs:** are often considered as "black boxes" because their inner workings can be challenging to interpret.

# CONVOLUTIONAL NEURAL NETWORKS

# (CNN/CONVNET)

# CONVOLUTIONAL NEURAL NETWORKS (CNN)

- **CNN** is a class of deep neural networks, most commonly applied to analyze visual imagery.



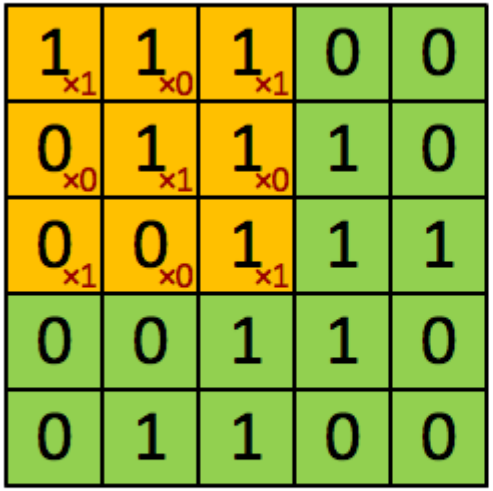
# CONVOLUTIONAL LAYER

- **Convolutional Layer**
- The convolution layer is the core building block of CNN.
- The main objective of convolution is to extract features such as edges, colors, corners from the input.
- As we go deeper inside the network,

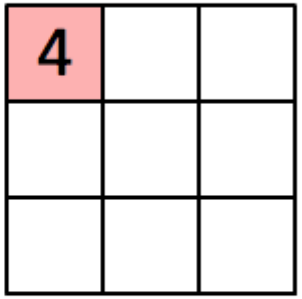
the network starts identifying more complex features such as shapes, digits, face parts as well.

$$O_h = \frac{I_h - k_h + 2P}{S_h} + 1$$

$$O_w = \frac{I_w - k_w + 2P}{S_w} + 1$$



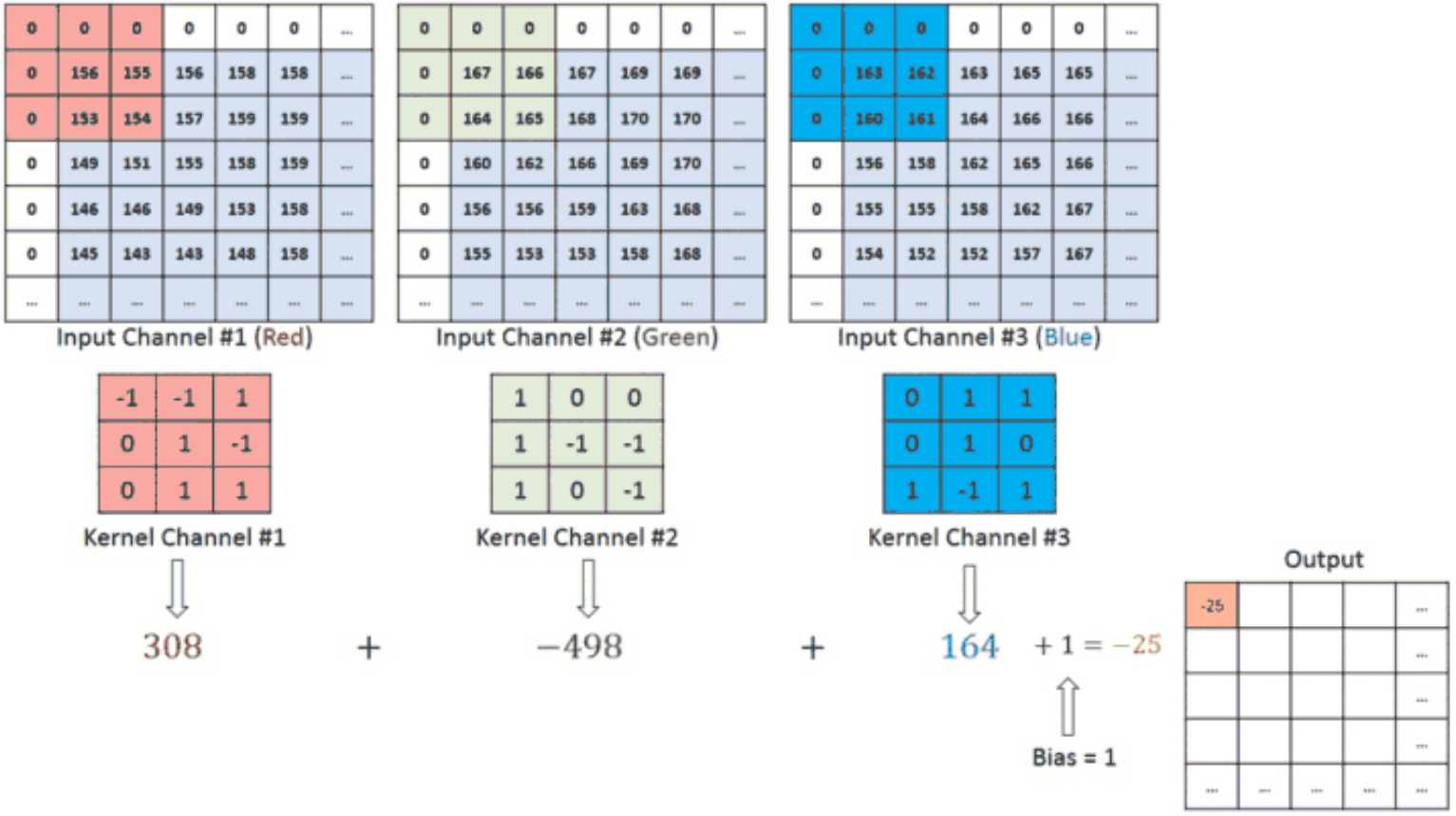

Image




Convolved Feature

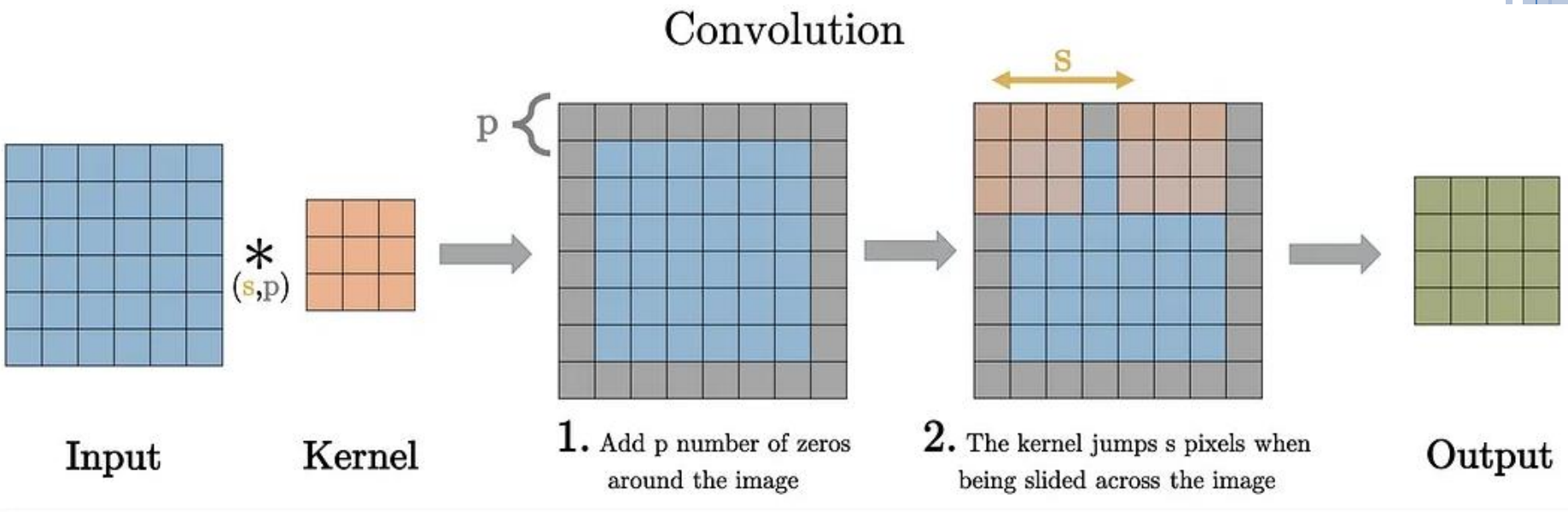
# CONVOLUTIONAL LAYER

- If the image is RGB



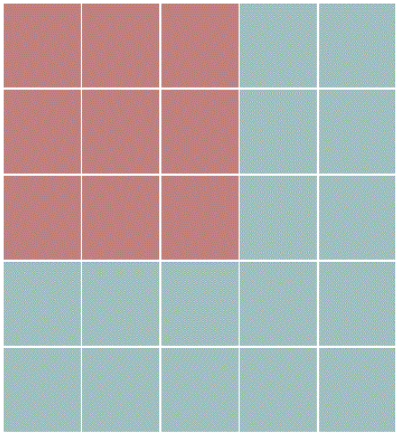
# CONVOLUTIONAL LAYER

- **Kernel Size:** defines the field of view of the convolution
- **Padding (p):** defines how the border of a sample is handled.
- **Stride:** defines the step size of the kernel when traversing the input image.

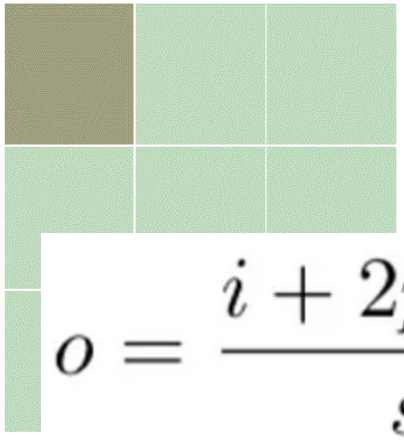


# CONVOLUTIONAL LAYER

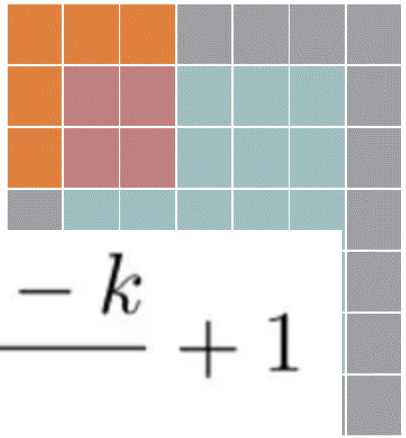
Type: conv - Stride: 1 Padding: 0

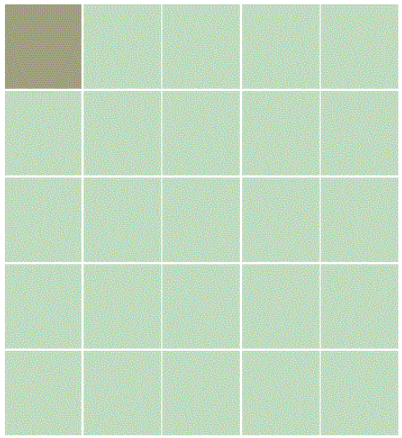


Input



Type: conv - Stride: 1 Padding: 1

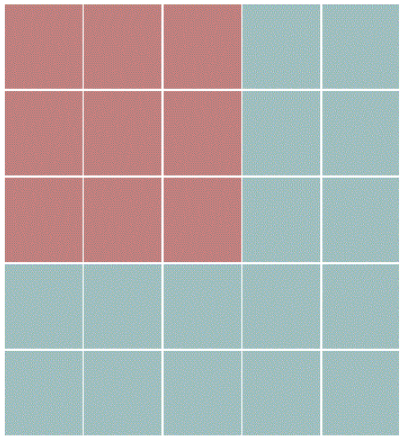




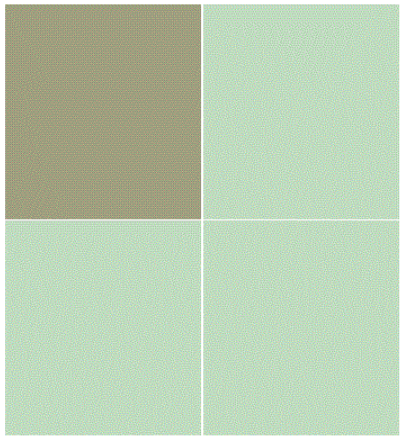
Output

$$o = \frac{i + 2p - k}{s} + 1$$

Type: conv - Stride: 2 Padding: 0

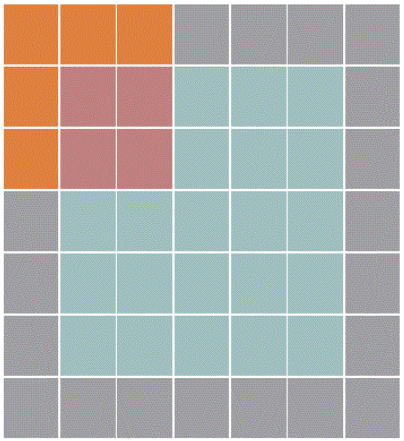


Input

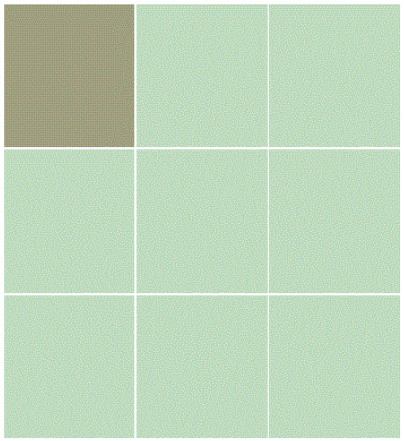


Output

Type: conv - Stride: 2 Padding: 1



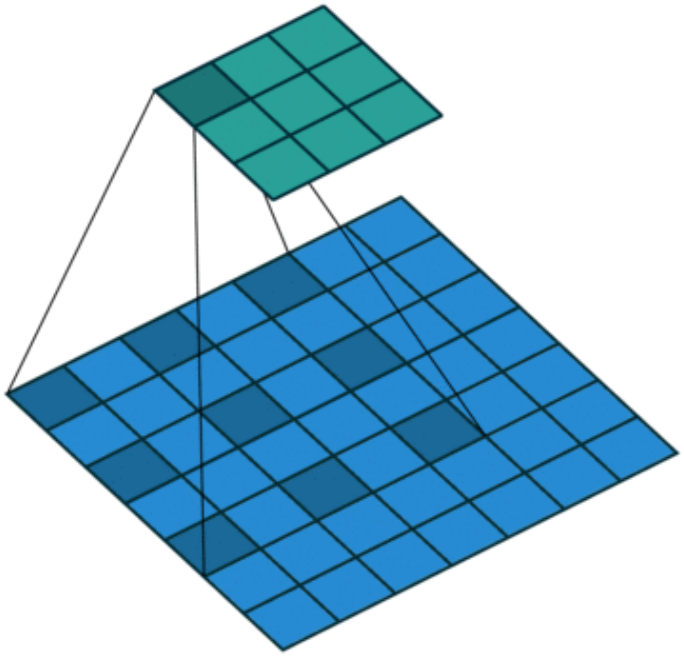
Input



Output

# CONVOLUTIONAL LAYER

- **Different Types of Convolutions in Deep Learning**
  - **Dilated Convolutions (**atrous convolutions)



2D convolution using 2 and no padding

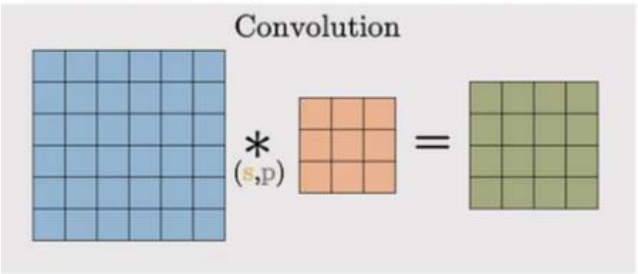
# CONVOLUTIONAL LAYER

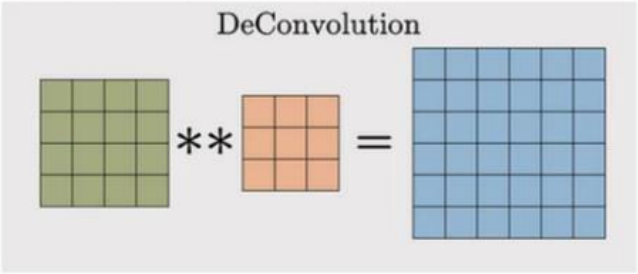
- **Different Types of Convolutions in Deep Learning**
  - **Transposed Convolutions:** it's not a deconvolution

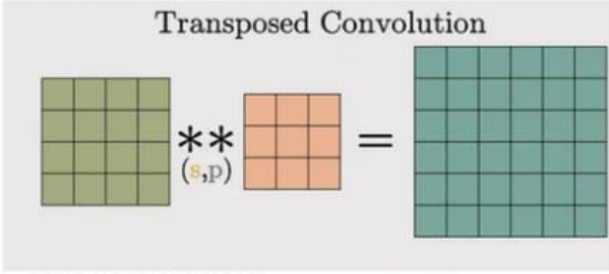
This is deconvolution:

# CONVOLUTIONAL LAYER

- **Different Types of Convolutions in Deep Learning**
- **Transposed Convolutions (TC)**
  - TC layer resembles the Deconvolutional layer as they both produce the same spatial dimension.
  - It is carried out for **Upsampling**.
  - it is defined by the **padding** and **stride**.

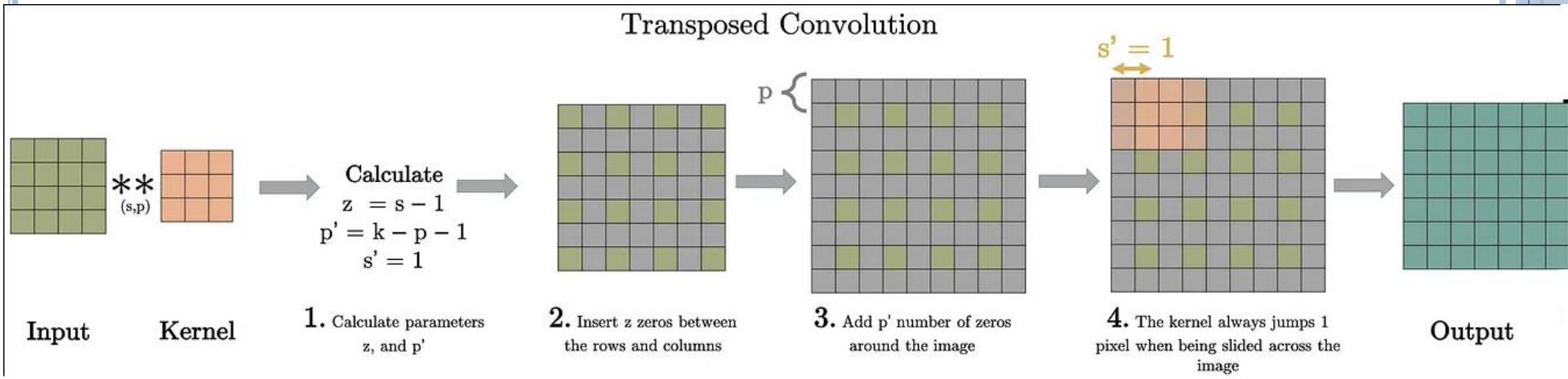


# CONVOLUTIONAL LAYER

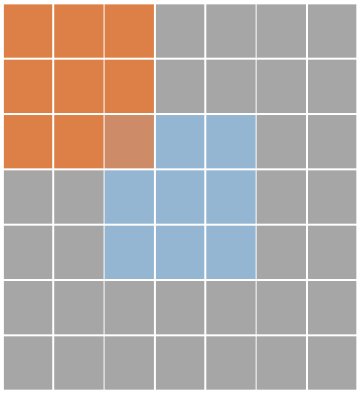
- **Different Types of Convolutions in Deep Learning**
- Implementing a **Transposed Convolutions** layer can be better explained as a 4 step process:



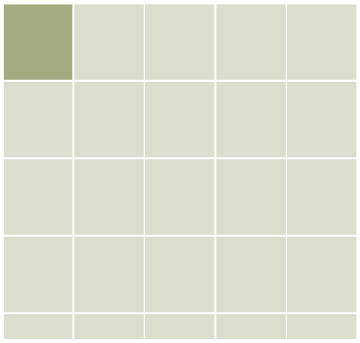
# CONVOLUTIONAL LAYER

- **Different Types of Convolutions in Deep Learning**
- **Transposed Convolutions**

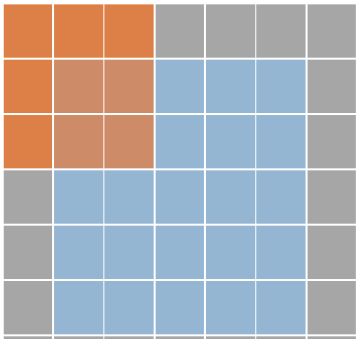
Type: transposed conv - Stride: 1 Padding: 0

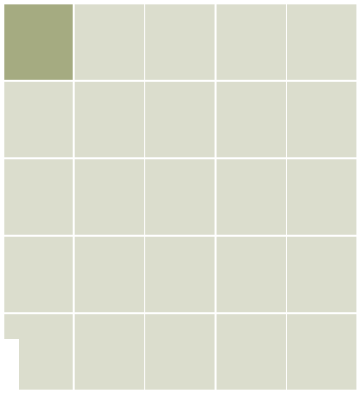


Input



Type: transposed conv - Stride: 1 Padding: 1

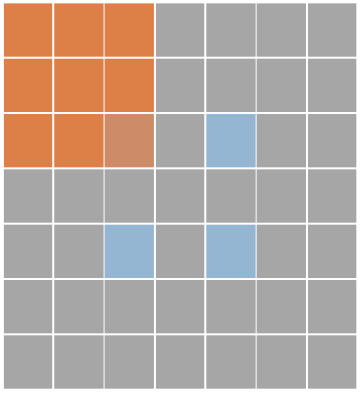




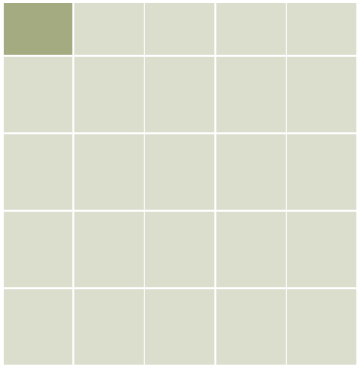
Output

$$o = (i - 1) \times s + k - 2p$$

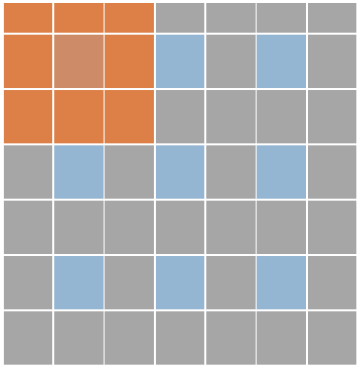
Type: transposed conv - Stride: 2 Padding: 1



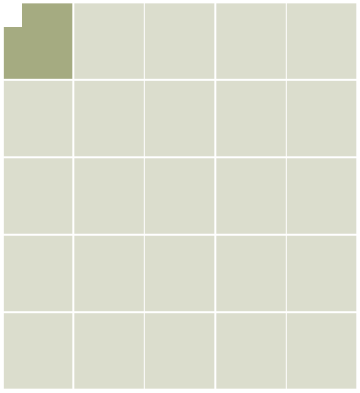
Input



Output



Input



Output

# CONVOLUTIONAL NEURAL NETWORKS

- **Different Types of Convolutions in Deep Learning**
  - **Transposed Convolutions "Summary"**

| Comparison | | | | | |
|---|---|---|---|---|---|
| Conv Type | Operation | Zero Insertions | Padding | Stride | Output Size |
| Standard | Downsampling | 0 | p | s | (i+2p-k)/s +1 |
| Transposed | Upsampling | (s - 1) | (k-p-1) | 1 | (i-1)*s+k-2p |

- The idea behind transposed convolution is to carry out trainable upsampling
- Transposed convolutions are standard convolutions but with a modified input feature map.
- The stride and padding **do not** correspond to the number of zeros added around the image and the amount of shift in the kernel when sliding it across the input, as they would in a standard convolution operation.

# CONVOLUTIONAL LAYER

- **Different Types of Convolutions in Deep Learning**
- **Separable Convolutions**
- we split the kernel operation into multiple steps
- convolution→ **y = conv(x, k)**
- where **y** is the output image, **x** is the input image, and **k** is the kernel.
- let's assume k can be calculated by: **k = k1.dot(k2)**.
- This would make it a separable convolution because instead of doing a 2D convolution with k, we could get to the same result by doing 2 1D convolutions with k1 and k2.

# CONVOLUTIONAL LAYER

- **Different Types of Convolutions in Deep Learning**
- **Separable Convolutions: Example**: Sobel kernel

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

x filter

| | | |
|---|---|---|
| +1 | +2 | +1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

y filter

Sobel X and Y filters

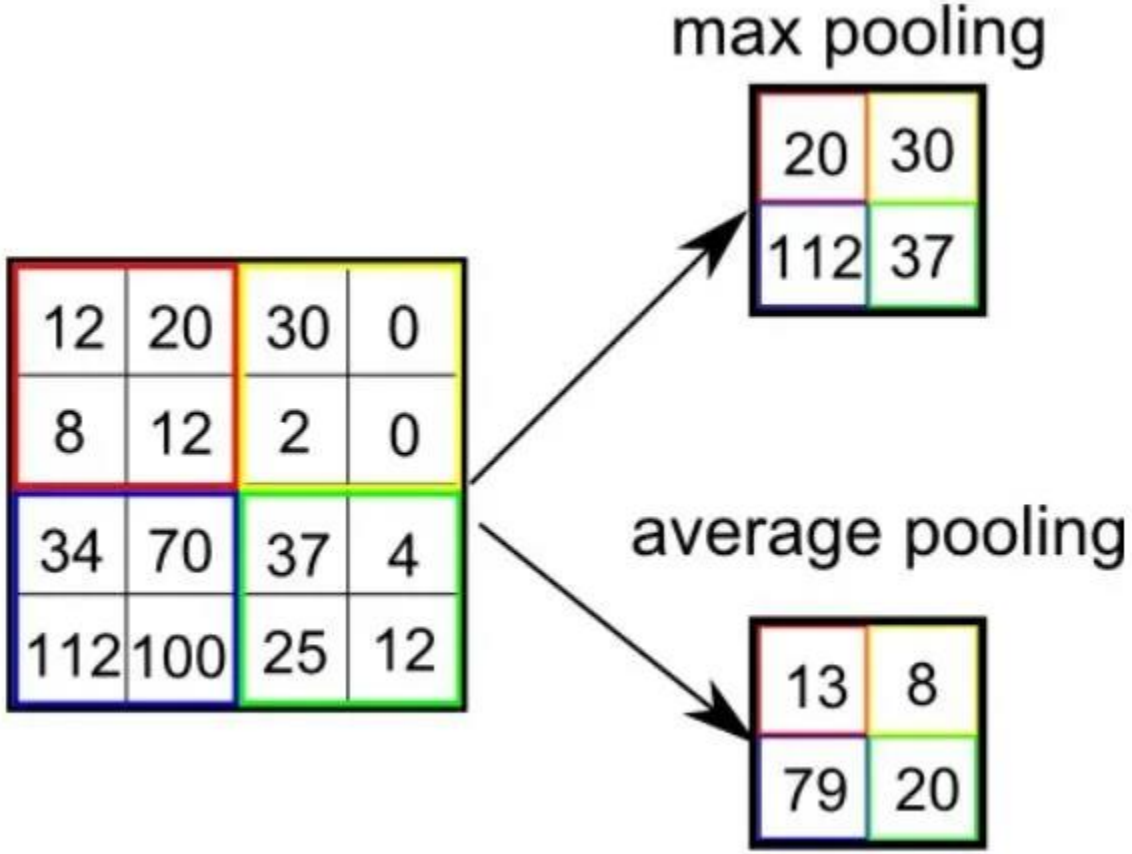
- You could get the same kernel by multiplying the vector [1, 0, -1] and [1,2,1]. T.
- 6 instead of 9 parameters while doing the same operation.
- This example shows what's called a **spatial separable convolution**.

# Pooling layer

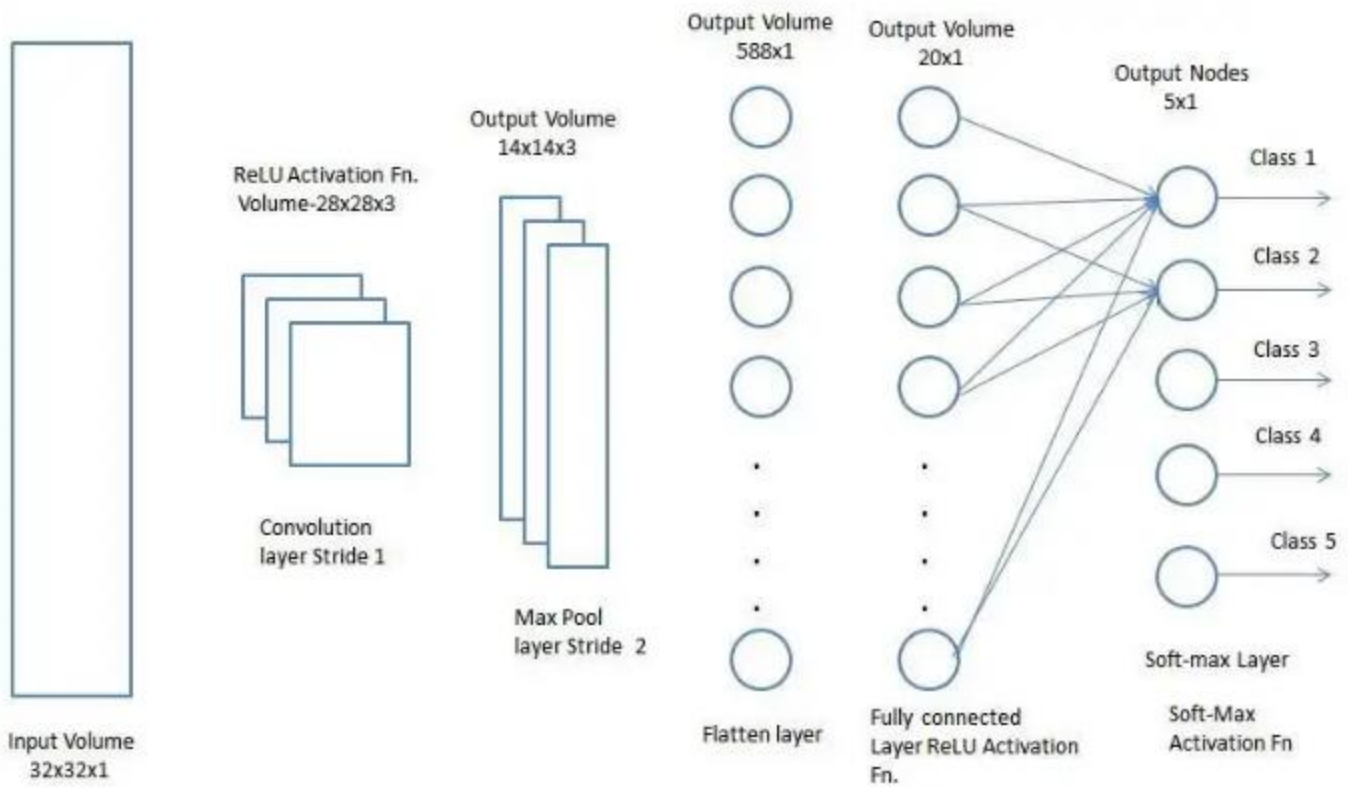
- **Pooling layer = Sub-sampling Layer**
- This layer decrease the computational power required to process the data.
- It is done by decreasing the dimensions of the featured matrix even more.
- It extract the dominant features from a restricted amount of neighborhood.


# Pooling layer

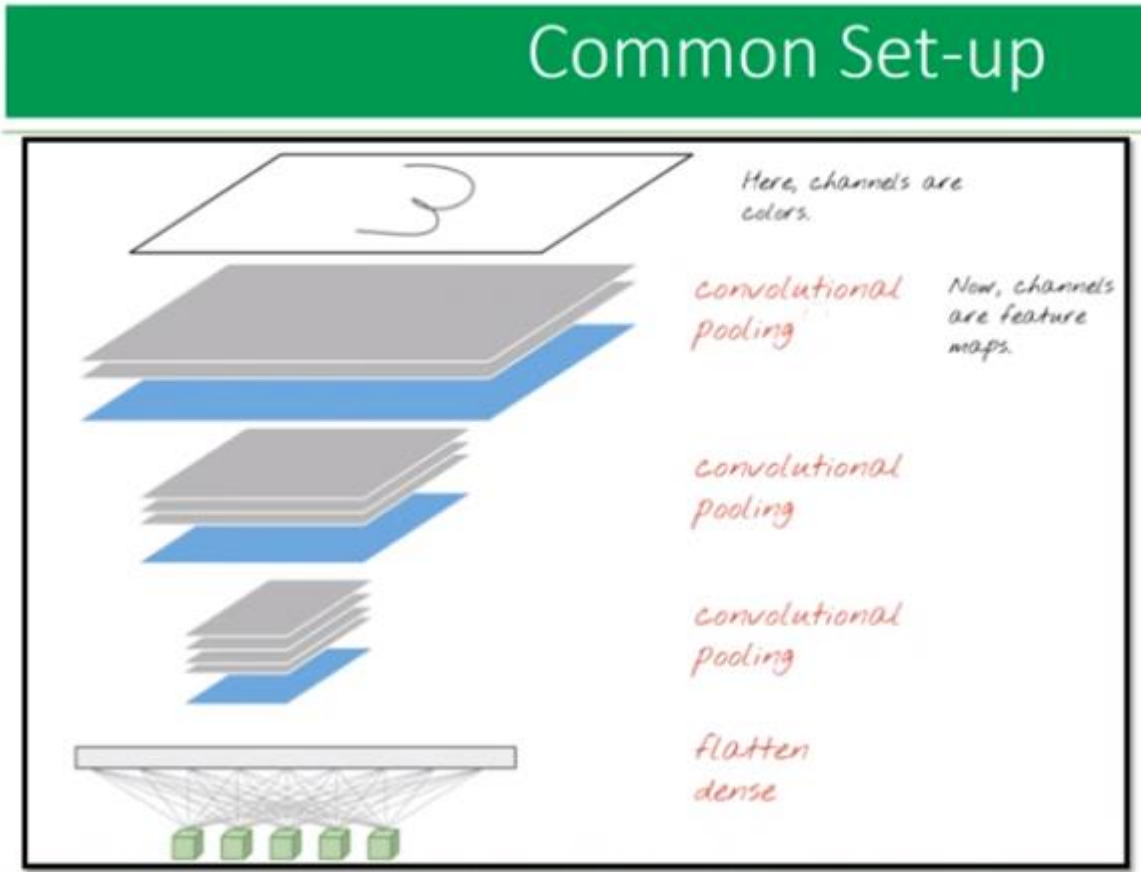
- **Pooling Layer types:**



# Fully Connected Layer (FCL)



# CONVOLUTIONAL NEURAL NETWORKS




In order to implement CNNs, most successful architecture uses one or more stacks of convolution + pool layers with relu activation, followed by a flatten layer then one or two dense layers.

# CONVOLUTIONAL NEURAL NETWORKS

- **Various architectures of CNNs**
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet
  - ResNet
  - ZFNet

# CONVOLUTIONAL NEURAL NETWORKS

## Summary

- Provide input image into convolution layer
- Choose parameters, apply filters with strides, padding if requires. Perform convolution on the image and apply ReLU activation to the matrix.
- Perform pooling to reduce dimensionality size
- Add as many convolutional layers until satisfied
- Flatten the output and feed into a FC Layer
- Output the class using an activation function (Logistic Regression with cost functions) and classifies images.

# CONVOLUTIONAL NEURAL NETWORKS

## CNN project 1: Cat Dog Recognition

```python
# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
# Initialising the CNN
classifier = Sequential()
# Step 1 - Convolution
classifier.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3),
                      activation = 'relu'))
# Step 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (2, 2)))
# Adding a second convolutional layer
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
# Step 3 - Flattening
classifier.add(Flatten())
# Step 4 - Full connection
classifier.add(Dense(units = 128, activation = 'relu'))
classifier.add(Dense(units = 1, activation = 'sigmoid'))
# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy',
                   metrics = ['accuracy'])
# Part 2 - Fitting the CNN to the images
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255,
shear_range = 0.2,
zoom_range = 0.2,
horizontal_flip = True)
```

Filters 32

Filter's shape

64x64 resolution and "3" stands for RGB

2x2 matrix

number of nodes

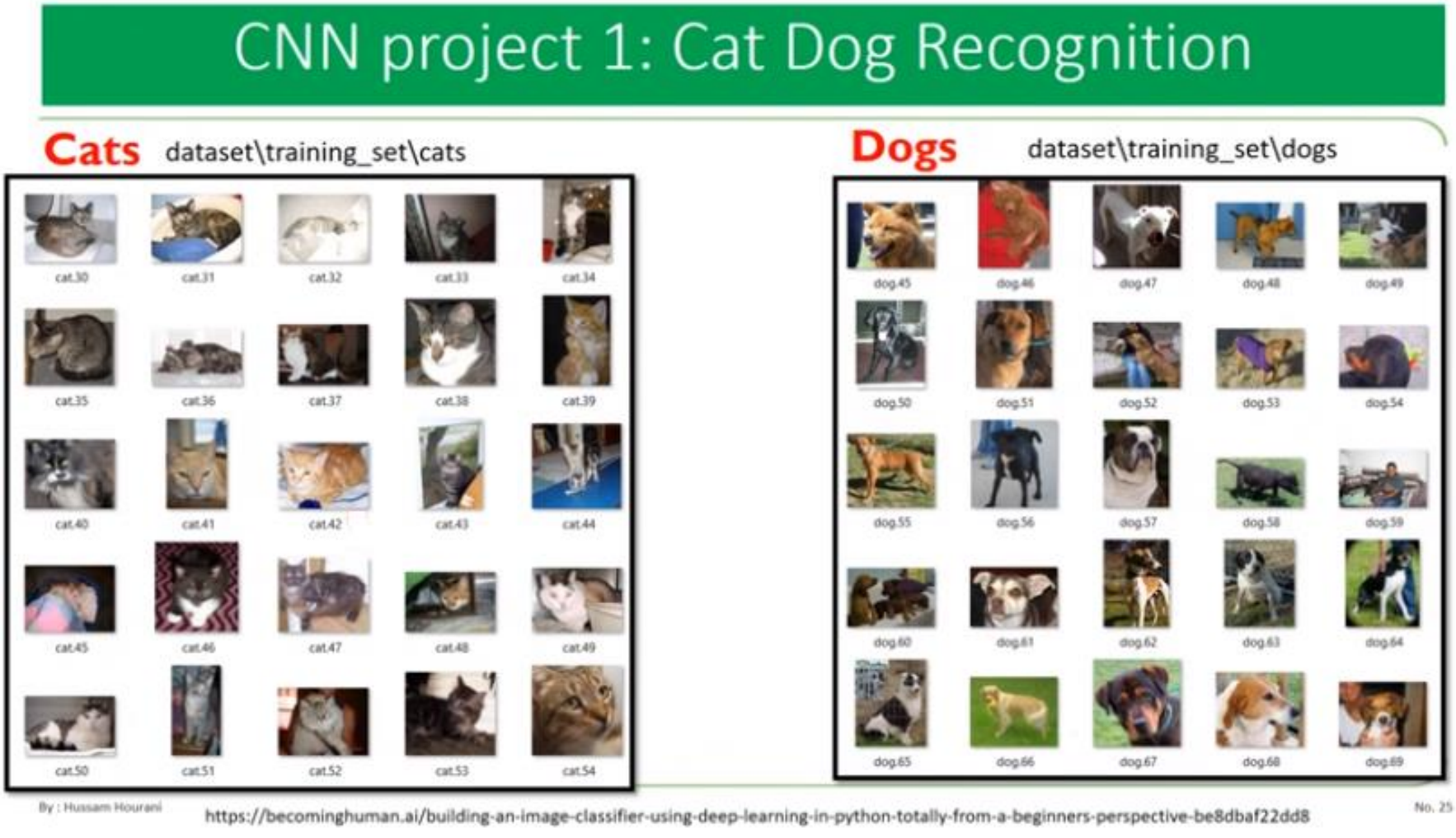
```python
test_datagen = ImageDataGenerator(rescale = 1./255)
training_set = train_datagen.flow_from_directory('dataset/training_set',
target_size = (64, 64),
batch_size = 32,
class_mode = 'binary')
test_set = test_datagen.flow_from_directory('dataset/test_set',
target_size = (64, 64),
batch_size = 32,
class_mode = 'binary')
classifier.fit_generator(training_set,
steps_per_epoch = 8000,
epochs = 25,
validation_data = test_set,
validation_steps = 2000)
# Part 3 - Making new predictions
import numpy as np
from keras.preprocessing import image
test_image = image.load_img('dataset/single_prediction/cat_or_dog_1.jpg',
                            target_size = (64, 64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = classifier.predict(test_image)
training_set.class_indices
if result[0][0] == 1:
    prediction = 'dog'
else:
    prediction = 'cat'
print(prediction)
```

By : Hussam Hourani

https://becominghuman.ai/building-an-image-classifier-using-deep-learning-in-python-totally-from-a-beginners-perspective-be8dbaf22dd8

https://github.com/venkateshtata/cnn_medium.

# CONVOLUTIONAL NEURAL NETWORKS



# CONVOLUTIONAL NEURAL NETWORKS

## CNN project 1: Cat Dog Recognition

```
In [1]: runfile('C:/Python36/Mystuff/Keras20_Conv.py', wdir='C:/Python36/Mystuff')
Using TensorFlow backend.
Found 2004 images belonging to 2 classes.
Found 198 images belonging to 2 classes.
Epoch 1/10
100/100 [==============================] - 194s 2s/step - loss: 0.6941 - acc: 0.5435 - val_loss: 0.6661 - val_acc: 0.6464
Epoch 2/10
100/100 [==============================] - 193s 2s/step - loss: 0.6532 - acc: 0.6213 - val_loss: 0.6158 - val_acc: 0.6767
Epoch 3/10
100/100 [==============================] - 191s 2s/step - loss: 0.6141 - acc: 0.6661 - val_loss: 0.5784 - val_acc: 0.7121
Epoch 4/10
100/100 [==============================] - 208s 2s/step - loss: 0.5658 - acc: 0.7094 - val_loss: 0.5674 - val_acc: 0.7323
Epoch 5/10
100/100 [==============================] - 188s 2s/step - loss: 0.5399 - acc: 0.7338 - val_loss: 0.5832 - val_acc: 0.7020
Epoch 6/10
100/100 [==============================] - 188s 2s/step - loss: 0.5215 - acc: 0.7348 - val_loss: 0.5224 - val_acc: 0.7474
Epoch 7/10
100/100 [==============================] - 186s 2s/step - loss: 0.4993 - acc: 0.7541 - val_loss: 0.5278 - val_acc: 0.7475
Epoch 8/10
100/100 [==============================] - 214s 2s/step - loss: 0.4792 - acc: 0.7757 - val_loss: 0.5346 - val_acc: 0.7575
Epoch 9/10
100/100 [==============================] - 212s 2s/step - loss: 0.4584 - acc: 0.7900 - val_loss: 0.5620 - val_acc: 0.7272
Epoch 10/10
100/100 [==============================] - 221s 2s/step - loss: 0.4379 - acc: 0.8060 - val_loss: 0.5833 - val_acc: 0.7122
dog
```

\dataset\single_prediction

# CONVOLUTIONAL NEURAL NETWORKS

## CNN project 2: Text Classification

```python
#https://realpython.com/python-keras-text-classification/
from keras.models import Sequential
from keras import layers
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer
import pandas as pd
from sklearn.model_selection import train_test_split

filepath_dict = {'yelp':   'yelp_labelled.txt',
                 'amazon': 'amazon_cells_labelled.txt',
                 'imdb':   'imdb_labelled.txt'}

df_list = []
for source, filepath in filepath_dict.items():
    df = pd.read_csv(filepath, names=['sentence', 'label'], sep='\t')
    df['source'] = source  # Add another column filled with the source
    df_list.append(df)

df = pd.concat(df_list)

df_yelp = df[df['source'] == 'yelp']

sentences = df_yelp['sentence'].values
y = df_yelp['label'].values

sentences_train, sentences_test, y_train, y_test = train_test_split(
    sentences, y, test_size=0.25, random_state=1000)

tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(sentences_train)

X_train = tokenizer.texts_to_sequences(sentences_train)
X_test = tokenizer.texts_to_sequences(sentences_test)
```

```python
vocab_size = len(tokenizer.word_index) + 1  # Adding 1 because of reserved 0

embedding_dim = 100
maxlen = 100

X_train = pad_sequences(X_train, padding='post', maxlen=maxlen)
X_test = pad_sequences(X_test, padding='post', maxlen=maxlen)

model = Sequential()
model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
model.add(layers.Conv1D(128, 5, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()

history = model.fit(X_train, y_train,
                    epochs=10,
                    verbose=False,
                    validation_data=(X_test, y_test),
                    batch_size=10)
loss, accuracy = model.evaluate(X_train, y_train, verbose=False)
print("Training Accuracy: {:.4f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
```

https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences

https://realpython.com/python-keras-text-classification/

# CONVOLUTIONAL NEURAL NETWORKS

## CNN project 2: Text Classification

**text files**

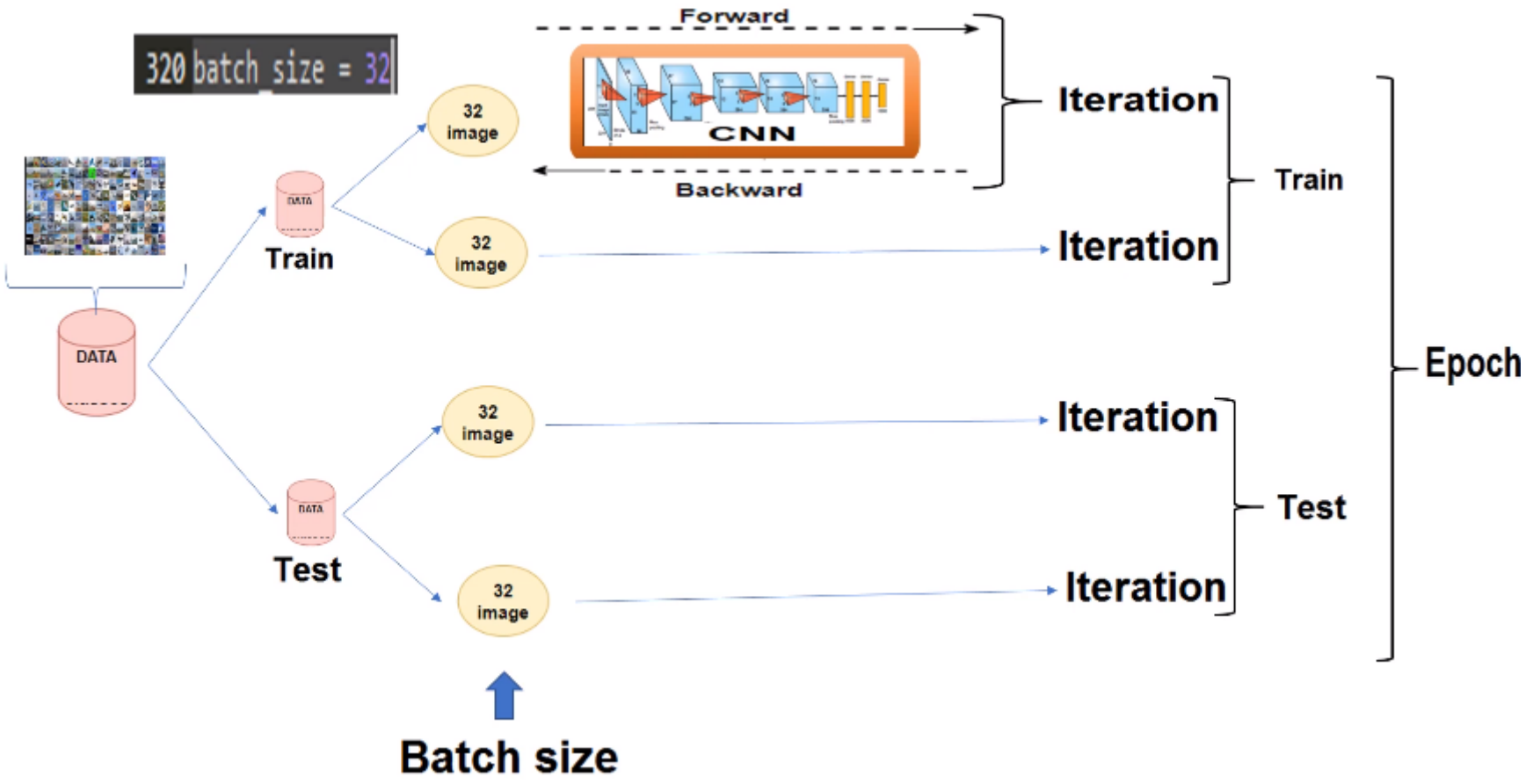
| | sentence | label | source |
|---|---|---|---|
| 0 | Wow... Loved this place. | 1 | yelp |
| 1 | Crust is not good. | 0 | yelp |
| 2 | Not tasty and the texture was just nasty. | 0 | yelp |
| 3 | Stopped by during the late May bank holiday of... | 1 | yelp |
| 4 | The selection on the menu was great and so wer... | 1 | yelp |
| 5 | Now I am getting angry and I want my damn pho. | 0 | yelp |
| 6 | Honeslty it didn't taste THAT fresh.) | 0 | yelp |
| 7 | The potatoes were like rubber and you could te... | 0 | yelp |
| 8 | The fries were great too. | 1 | yelp |
| 9 | A great touch. | 1 | yelp |
| 10 | Service was very prompt. | 1 | yelp |
| 11 | Would not go back. | 0 | yelp |
| 12 | The cashier had no care what so ever on what I... | 0 | yelp |
| 13 | I tried the Cape Cod ravoli, chicken,with cran... | 1 | yelp |
| 14 | I was disgusted because I was pretty sure that... | 0 | yelp |
| 15 | I was shocked because no signs indicate cash o... | 0 | yelp |
| 16 | Highly recommended. | 1 | yelp |
| 17 | Waitress was a little slow in service. | 0 | yelp |
| 18 | This place is not worth your time, let alone V... | 0 | yelp |
| 19 | did not like at all. | 0 | yelp |
| 20 | The Burrittos Blah! | 0 | yelp |
| 21 | The food, amazing. | 1 | yelp |
| 22 | Service is also cute. | 1 | yelp |
| 23 | I could care less... The interior is just beau... | 1 | yelp |
| 24 | So they performed. | 1 | yelp |

```
Name: 0, dtype: object
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 100)          174700
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 96, 128)           64128
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 240,129
Trainable params: 240,129
Non-trainable params: 0
_________________________________________________________________
WARNING:tensorflow:From C:\Anaconda37\lib\site-packages\tensorflow\python
\ops\math_grad.py:102: div (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Deprecated in favor of operator or tf.math.divide.
Training Accuracy: 1.0000
Testing Accuracy:  0.7840
```

# CONVOLUTIONAL NEURAL NETWORKS

- **Key Differences between MLP and CNN**
- CNN is mostly used for Image Data, whereas it is better to use MLP on structural data;
- CNN has less parameters and tries to reduce the dimensions of image whereas in case of MLP number of parameters depends on the data;
- CNN is complex in nature whereas MLP is relatively simple compared to CNN;
- CNN uses special Convolution and Pooling Layers whereas MLP is just a network of Neurons;
- CNN is generally used for huge data as compared to MLP

# DL: Key Concepts



# DL: Key Concepts

- **Worked Example**

- Assume you have a dataset with 200 samples and you choose a batch size of 5 and 1,000 epochs.
- This means that the dataset will be divided into 40 batches, each with 5 samples. The model weights will be updated after each batch of 5 samples.
- This also means that one epoch will involve 40 batches or 40 updates to the model.
- With 1,000 epochs, the model will be exposed to or pass through the whole dataset 1,000 times. That is a total of 40,000 batches during the entire training process.

# DL: Key Concepts

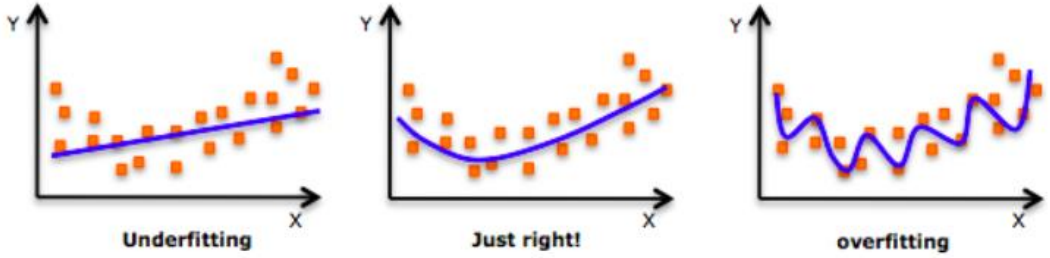
- **What is Softmax in deep learning?**

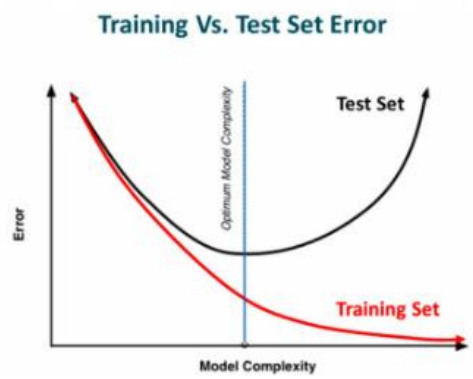
  - Softmax assigns decimal probabilities to each class in a multi-class problem.

  - Those decimal probabilities must add up to 1.0.

# DL: Key Techniques

- **Regularization in DL**
- In deep learning, regularization is a technique used to prevent overfitting.
- It makes slight modifications to the learning algorithm such that the model generalizes better.
- This in turn improves the model's performance on the unseen data as well.



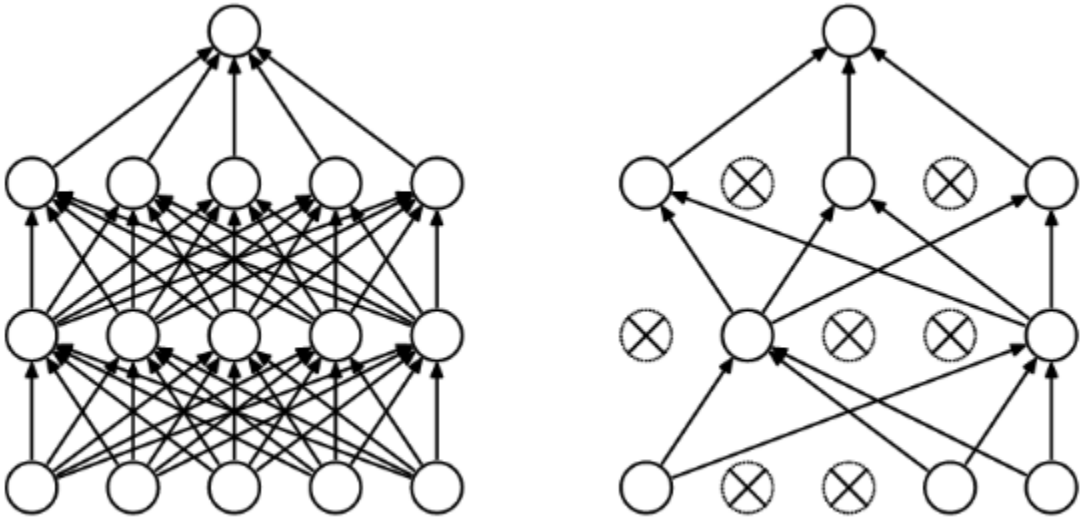



# DL: Key Techniques

- **Regularization**
  - Different Regularization Techniques in Deep Learning
    1. Dropout
    2. Data augmentation
    3. Early stopping
    4. L2 and L1 regularization

# DL: Key Techniques

- **What does dropout do in neural network?**



(a) Standard Neural Net

(b) After applying dropout.

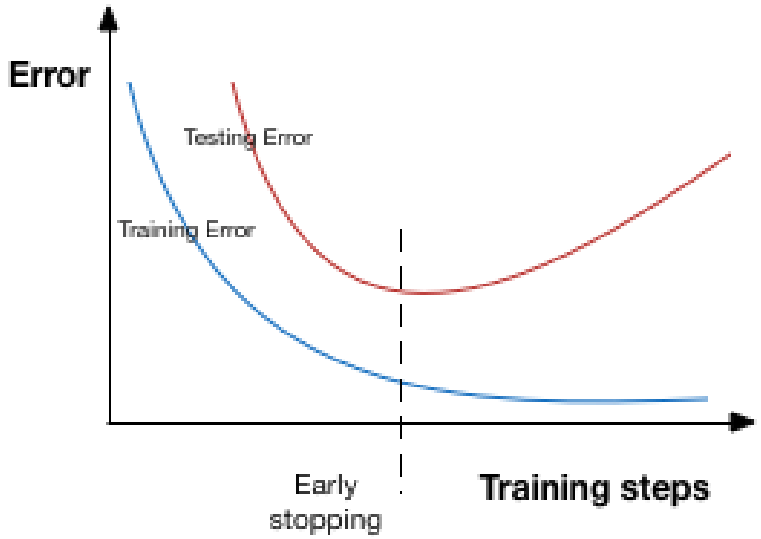
# DL: Key Techniques

- **Data Augmentation**
- The simplest way to reduce overfitting is to increase the size of the training data.

# DL: Key Techniques

## Early stopping

- In Early stopping, we keep one part of the training set as the validation set.
- When we see that the performance on the validation set is getting worse, we immediately stop the training on the model.



# DL: Key Techniques

- **Batch normalization**
  - it is a technique that normalizes the input of each layer within a mini-batch during training.
  - it accelerates training, makes convergence more stable, and reduces the sensitivity to weight initialization.
  - It enables the use of higher learning rates and often results in better generalization.
  - It is implemented as a layer within a neural network.

# DL: Key Concepts

- **Transfer learning ?**
- It is a technique where a pre-trained NN is used as a starting point for a different but related task.

- Instead of training a new network from scratch, you fine-tune the pre-trained model to adapt to your specific problem by adjusting its final layers