

CNN FOR DUMMIES



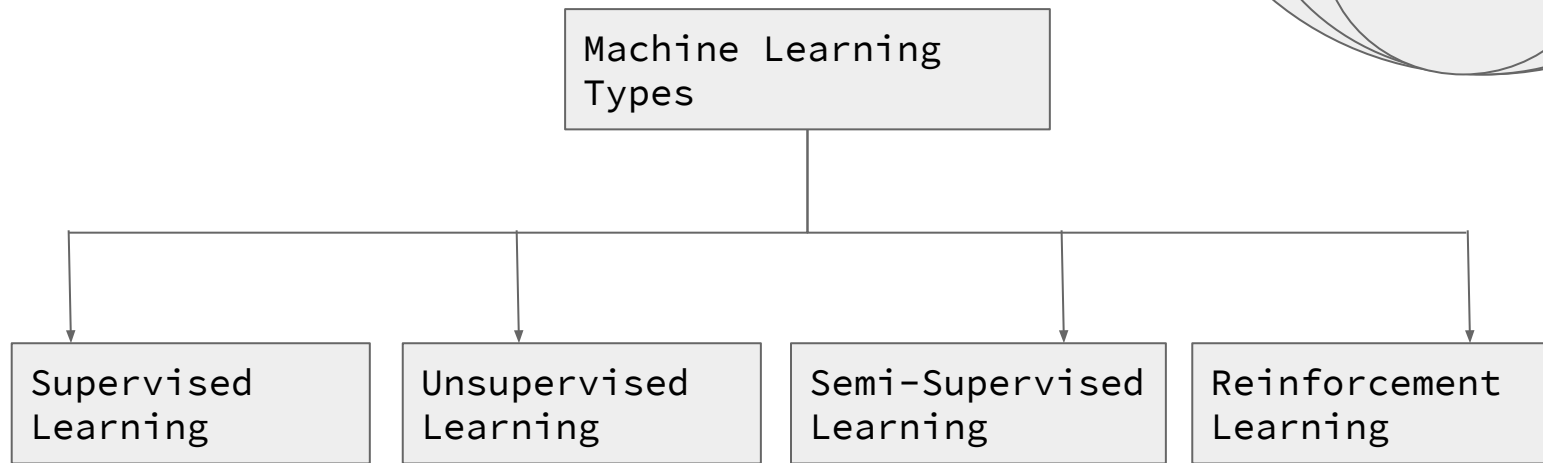
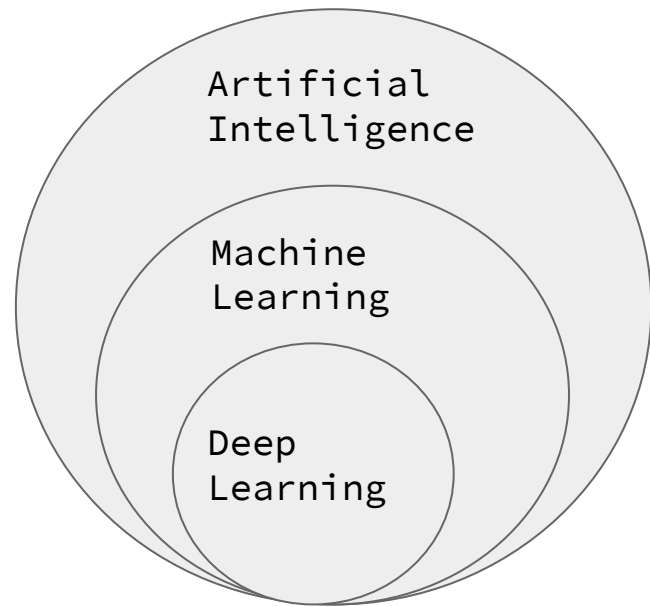
Mounika Vanka
Staff Researcher
Lenovo Research

Rajanie Prabha
ML Research Scientist
Pittsburgh Supercomputing
Center

AGENDA

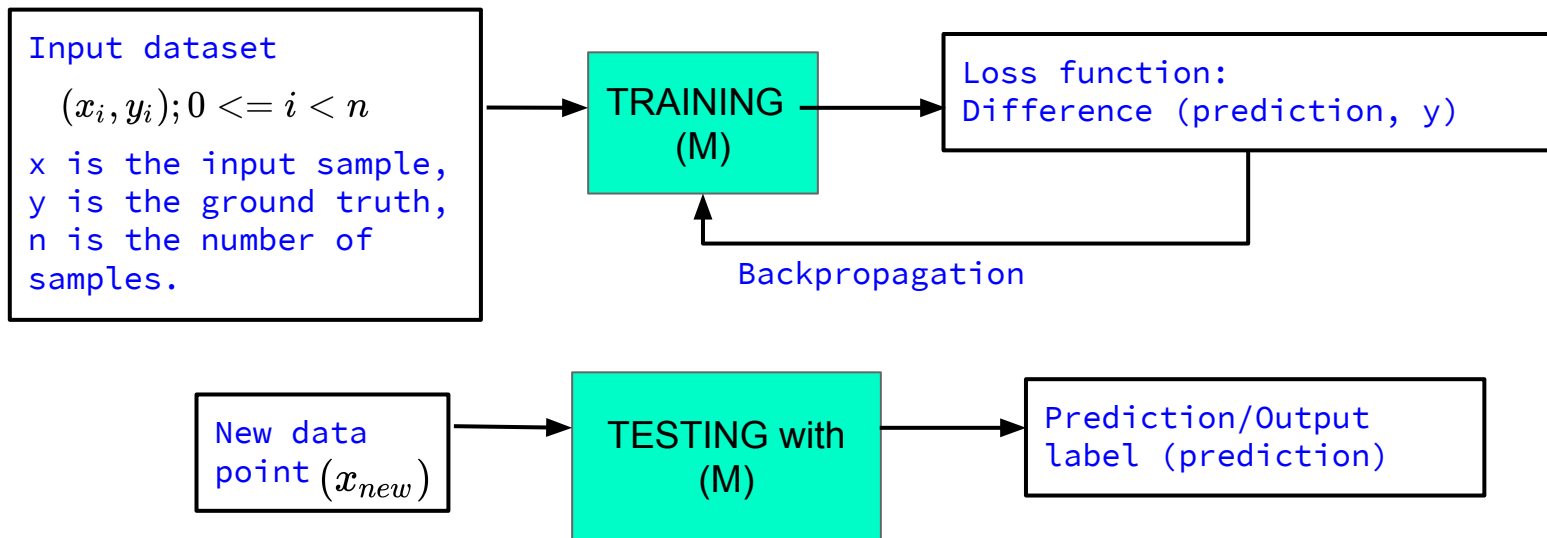
- Machine Learning Algorithms
- What is Deep Learning (DL)?
- Deep Learning Applications
- What do we want to do? Image Classification
- Why not MLPs?
- How to solve this task?
- What exactly are CNNs?
- Forward Pass and Backward Pass
- Pooling layer
- Activation functions and loss functions
- Generalization, overfitting and underfitting
- Splitting, k-fold validation
- Hyperparameters
- Advantages and Challenges

MACHINE LEARNING ALGORITHMS

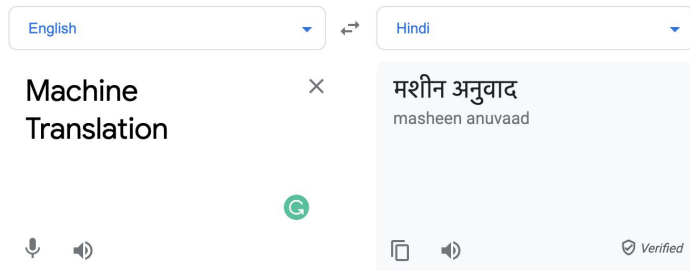
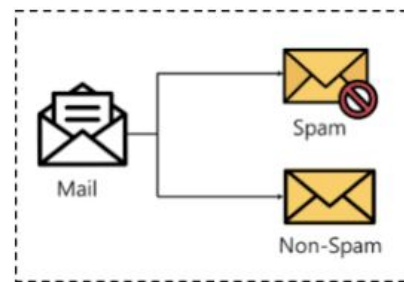
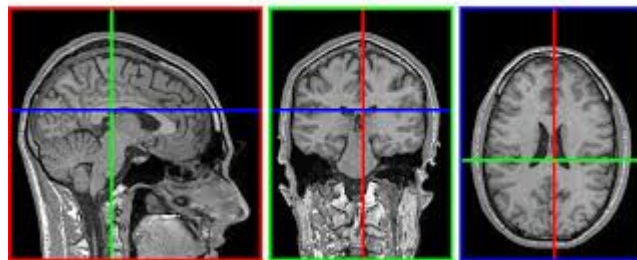


WHAT IS DEEP LEARNING (DL)?

The idea of DL is to learn patterns and representations from the dataset to predict labels for future samples.



DEEP LEARNING APPLICATIONS



WHAT DO WE WANT TO DO?: IMAGE CLASSIFICATION TASK



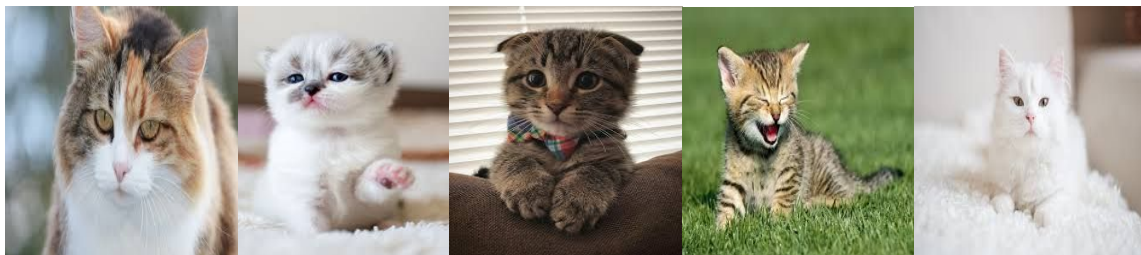
Input image

Feature learning
(Model training)



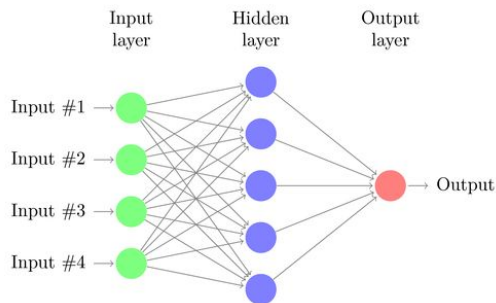
Prediction: Cat
Probability: 0.8

The model needs
to learn cat
features.



MLPS? WHY NOT?

Traditional NNs called Multilayer Perceptron (MLP)



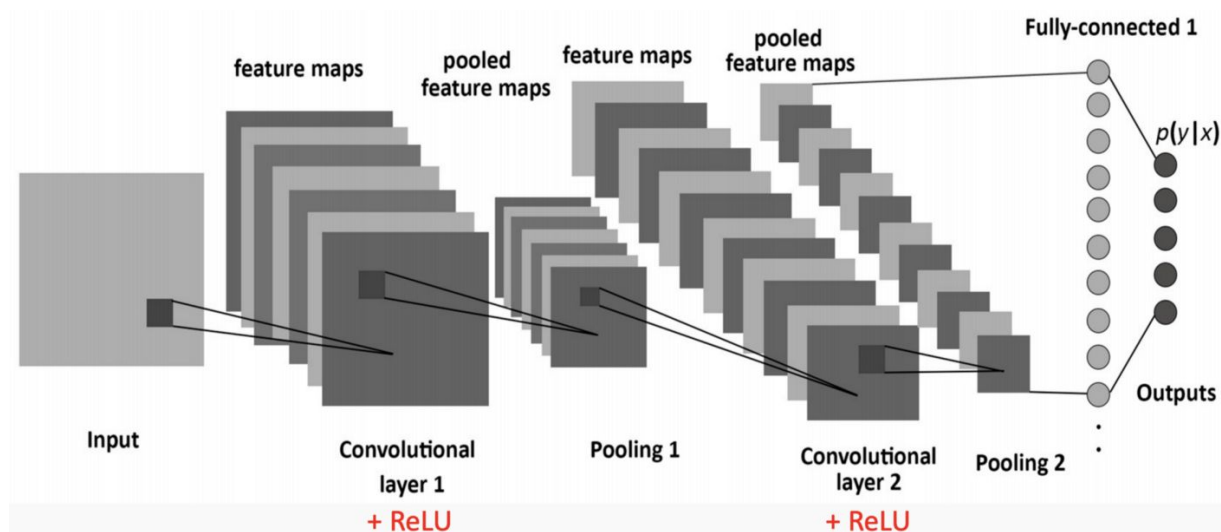
$$y = f(x; \theta)$$

Where θ is the set of model parameters.

- MLPs use one perceptron for each input (e.g. pixel in an image, multiplied by 3 in RGB case). The amount of weights rapidly becomes unmanageable for large images. For a 224 x 224 pixel image with 3 color channels there are around 150,000 weights that must be trained!
- They are not translation invariant.
- Spatial information is lost when the image is flattened into an MLP.

HOW TO SOLVE THIS TASK?- CNNs

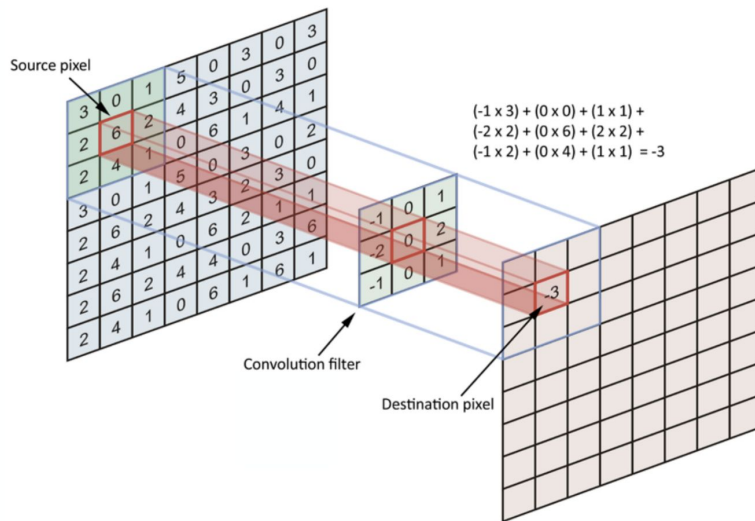
- The idea is to learn representations from the data.
- Earlier, these representations were learnt from hand-crafted features.



OKAY, SO WHAT EXACTLY ARE CNNs?

In order to overcome the various drawbacks of MLPs, Convolutional Neural Networks came into picture. Intuitively,

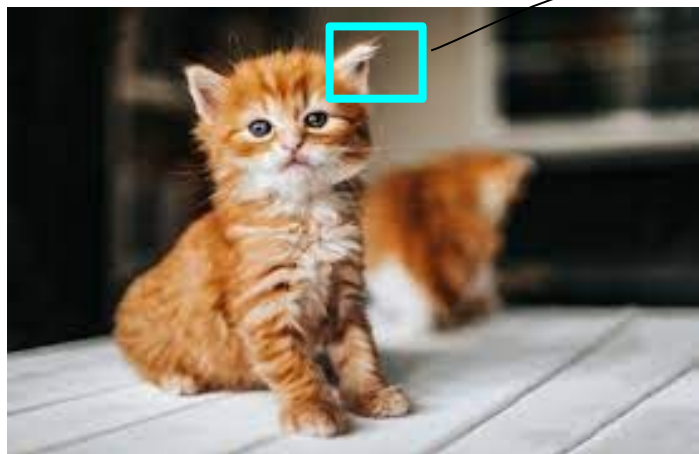
Destination pixel = $\text{Sum}(\text{Product}(\text{Source pixel}, \text{Convolution filter}))$



Apply it on all the pixels of all the channels.

CNNS- CONTINUED

- The convolution filters most popularly used are 2x2, 3x3, 5x5. They are smaller than the whole image size (let's say 224x224) because nearby pixels are more strongly related than distant ones.
- Objects are build up out of smaller parts (features).



```
00000000000000000000
00001111100000000000
11110001000000000000
00000001000000000000
00000001000000000000
```



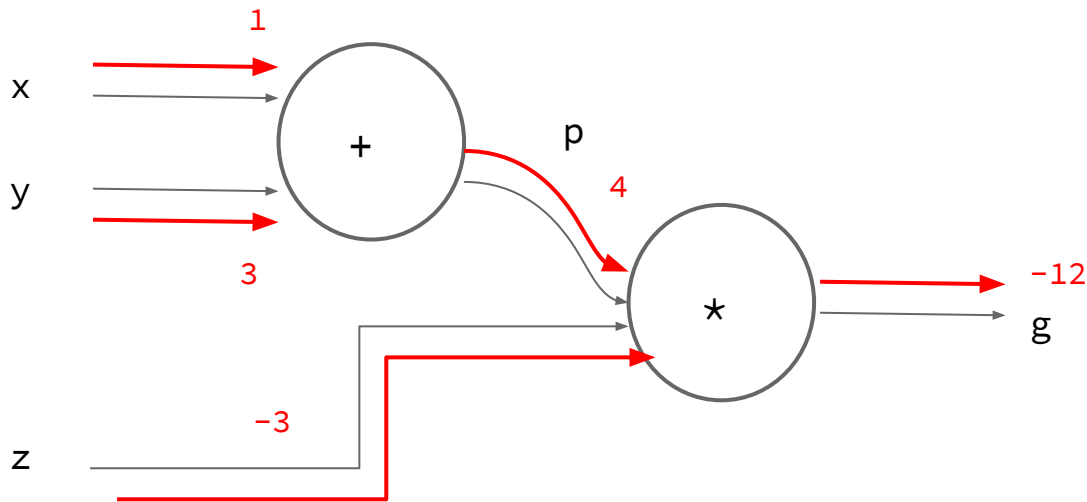
FORWARD PASS AND BACKWARD PASS: THE MATH BEHIND

Imagine the equation as a computational graph.

Let's say $g = (x+y)*z$

The graph would be:

Let's say $x=1$, $y=3$, $z=-3$
→ Forward pass



BACKWARD PASS: THE MATH BEHIND

In the backward pass, our intention is to compute the gradients for each input with respect to the final output and then we use these gradients to update the weights.

Let's say $x=1$, $y=3$, $z=-3$

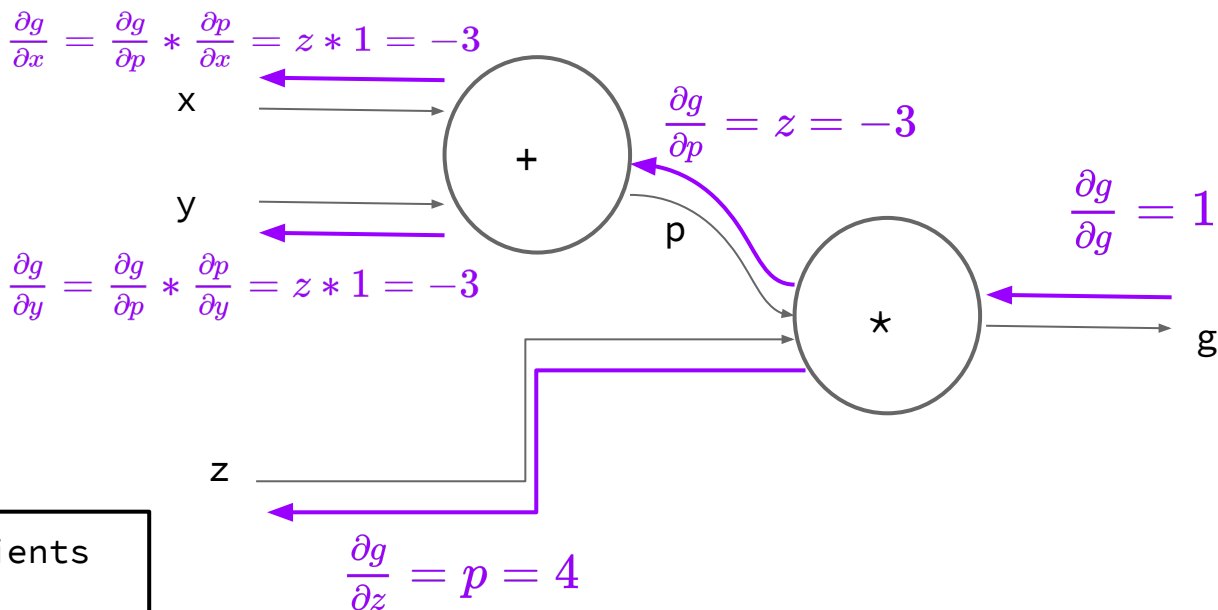
← Backward pass

From the forward pass,
we know the value of $p=4$

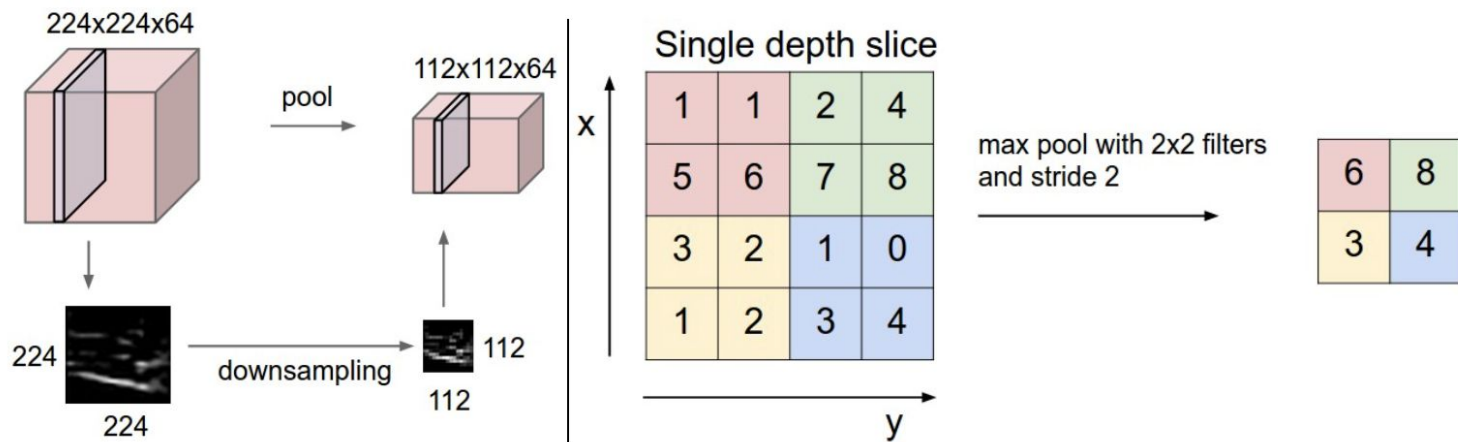
$$g = (x+y)*z$$

$$p = (x+y)$$

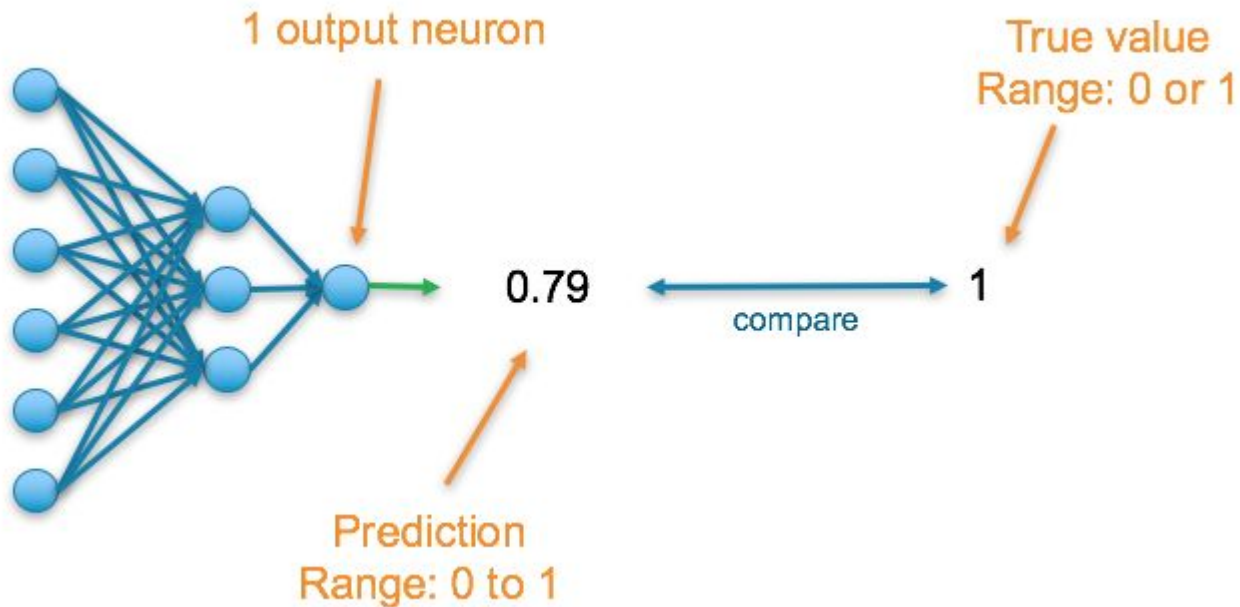
$$W_{\text{new}} = W_{\text{old}} - \text{Learning Rate} * \text{gradients}$$



POOLING LAYER

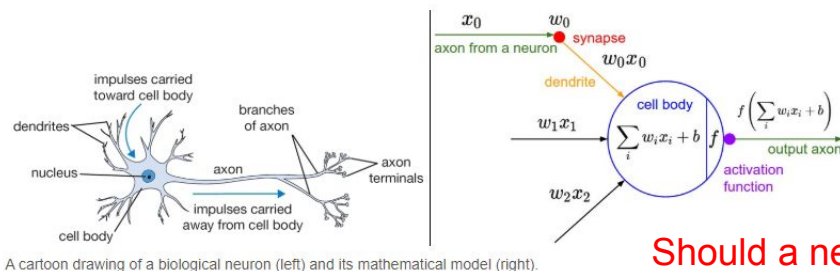


ACTIVATION AND LOSS FUNCTIONS



ACTIVATION FUNCTION

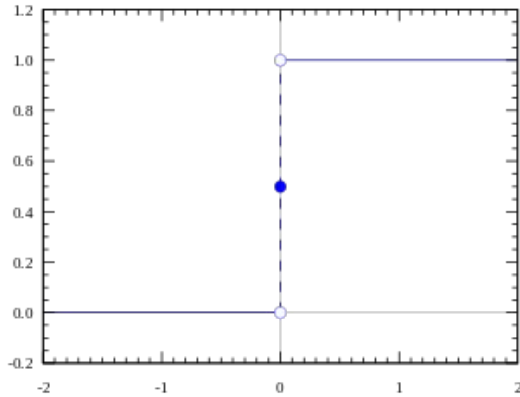
- Determines what the output looks like for a given input
- Biologically inspired from the activity of the neurons
- As seen from a NN, the weighted inputs are transformed to an output value



Should a neuron fire or not?

- As an example, when you smell something pleasant a particular set of neurons are activated, and when you smell something unpleasant, another set of neurons are activated

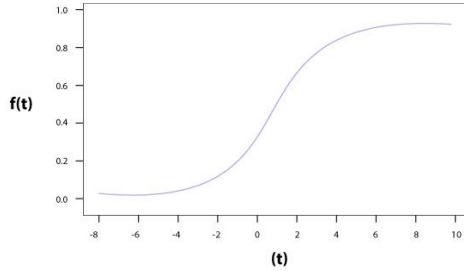
STEP FUNCTION



- Activation function is activated when $y > \text{threshold}$ and the output is either 0 or 1
- Okay, so what do you do when you have multiple Classes to be classified (Cat, Dog, Mouse, etc.)

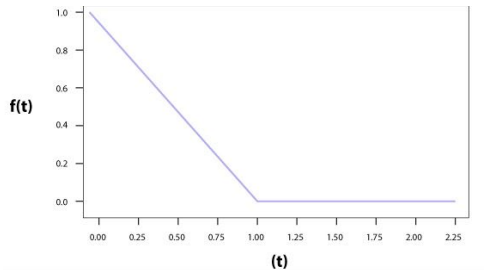
OTHER TYPES:

Sigmoid



Range is between 0 and 1







"Relu" – Rectified Linear



All negative input -> non zero value

LOSS FUNCTION

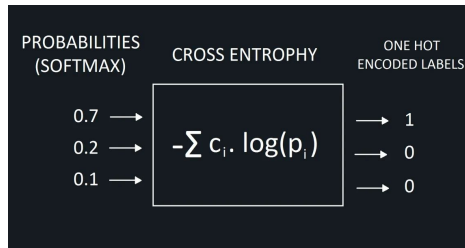
Error - Difference between what the output should be to what it actually is?

OUTPUT	CORRECT VALUE	OBJECTIVE $F(X)$	VALUE
		far from reality	200
		closer	100
		very close	0

TYPES OF LOSS FUNCTIONS

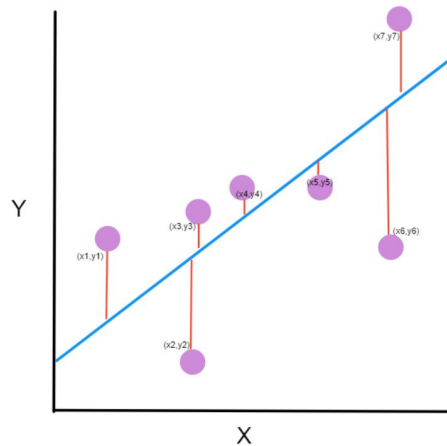
Cross Entropy:

- Default loss function for classification problems (0 or 1)
- Score based on the average difference between actual and predicted probability distribution
- Perfect cross entropy value is 0



Mean Squared Error:

- Default error for regression problems (real valued quantity)
- Average of squared difference
- Larger mistakes result in more error due to squaring
- Always positive



EXAMPLE PROBLEM, ACTIVATION AND LOSS FUNCTIONS

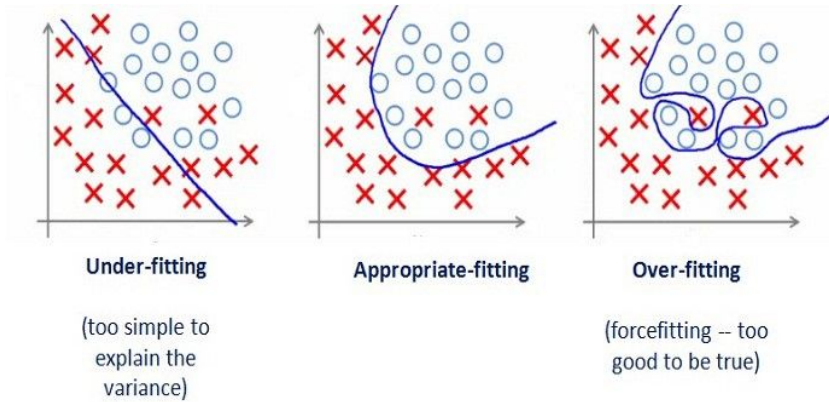
<u>Problem</u>	<u>Output Type</u>	<u>Activation Function</u>	<u>Loss Function</u>
Regression	Numerical	Linear	Mean Squared Error
Classification	Binary	Sigmoid	Binary Cross Entropy
Classification	Single Label, Multiple Class	Softmax (probability)	Cross Entropy

GENERALIZATION, OVERFITTING AND UNDERFITTING

- Generalization? What is it?
 - How well the concepts learned by a machine learning model apply to specific examples not seen by the model when it was learning
- Make predictions in the future on data never seen
- Better generalization, better the model

Two biggest causes of poor performance of a model:

- Underfitting
- Overfitting



• Underfitting:

- Model can neither learn the training data nor the generalize to new data

■ Solution:

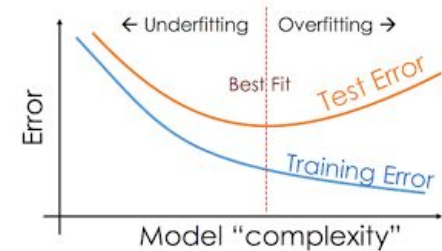
- Try alternate algorithms
- Increasing the size of the training dataset

• Overfitting:

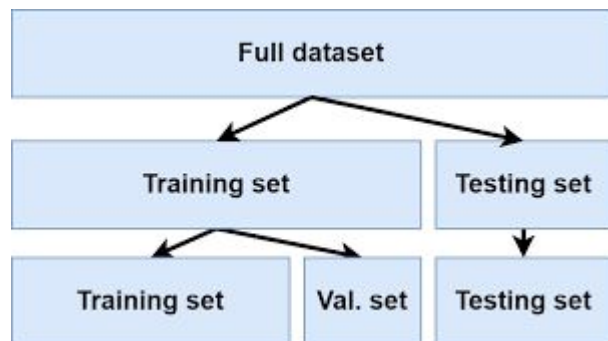
- Model fits the training data too well
- Learns so well that the details including noise are learnt
- More common than underfitting

■ Solution:

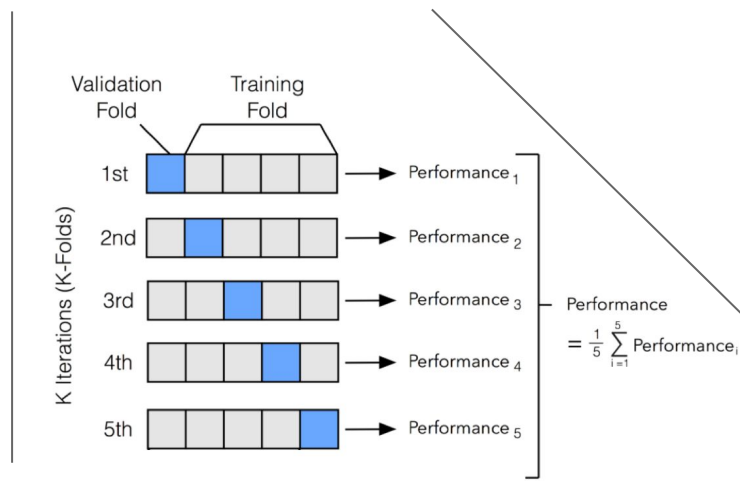
- Hold back a validation set (data not seen during training)
- Resampling technique (k-fold validation)
- Regularization
- Data augmentation techniques (rotation, translation, flip, etc)



SPLITTING THE DATASETS



HOLDING BACK THE DATA



K-FOLD CROSS VALIDATION

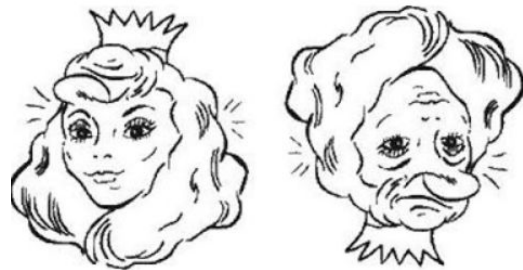
HYPERPARAMETERS

- Value external to the model and cannot be estimated from the data (knobs or dials of the model training process)
- Estimate model parameters (Manual search, Random search, Grid search, etc)
- Often set as heuristics
- Examples:
 - Learning rate (α)
 - Number of epochs
 - Batch size

ADVANTAGES AND DISADVANTAGES

Advantages:

- Automatically detects important features without human supervision
- Ability to deliver high-quality results
- Unsupervised methods do not require data labeling



Challenges:

- Computationally expensive and requires huge datasets
- Generating a supervised data is expensive and sometimes not feasible
- Adversarial example (modified images with noise)
- Coordinate Frame (orientation and features)

GOOGLE CO-LAB EXERCISE

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

← Packages to be used

```
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()
train_images, test_images = train_images / 255.0, test_images / 255.0
```

Collect and preprocess

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

Create a convolutional base model

```
model.compile(optimizer='adam',  
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),  
              metrics=['accuracy'])  
  
history = model.fit(train_images, train_labels, epochs=10,  
                    validation_data=(test_images, test_labels))
```

Train the model

```
plt.plot(history.history['accuracy'], label='accuracy')  
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')  
plt.xlabel('Epoch')  
plt.ylabel('Accuracy')  
plt.ylim([0.5, 1])  
plt.legend(loc='lower right')  
  
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
```

Evaluate model

REFERENCES

<https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>

<https://machinelearningmastery.com/cross-entropy-for-machine-learning/>

<https://towardsdatascience.com/regularization-an-important-concept-in-machine-learning-5891628907ea>

<https://www.analyticssteps.com/blogs/how-decide-which-activation-function-and-loss-function-use>

<https://www.analyticssteps.com/blogs/7-types-activation-functions-neural-network>

<https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>

<https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/>

<https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/>

<https://medium.com/ml-research-lab/under-fitting-over-fitting-and-its-solution-dc6191e34250>

<https://iq.opengenus.org/disadvantages-of-cnn/>

https://medium.com/@AI_with_Kain/understanding-of-multilayer-perceptron-mlp-8f179c4a135f

<https://lipyeow.github.io/ics491f17/morea/deepnn/ImageClassification-CNN.pdf>

<https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253>

https://www.youtube.com/watch?v=IVVVjBSk9N0&ab_channel=SirajRaval

https://www.tutorialspoint.com/python_deep_learning/python_deep_learning_computational_graphs.htm