PyTorch and CNNs

Agenda

1. Overview of AI and Deep Learning (17.15-17.25)
2. Datasets (17.25-17.30)
    a. Task: Implement Dataset in Pytorch (17.30-17.40)
    b. Solution (17.40-17.45)
3. CNNs
    a. Convolution (17.45-18.00). Task (18.15-18.25). Solution (18.25-18.30)
    b. Other Layers (18.30-18.35). Task (18.35 - 1845). Solution (1845-1850) (DROP)
4. Training Loop (18.50 - 19.00). Task (19.00-19.15). Solution (19.15-1920)
5. Specific CNN Architectures (19.20-19.25).
6. Competition: Create the most accuracy custom CNN! (19.25-20.00)

HELLO!

I AM ULRIK RØSBY

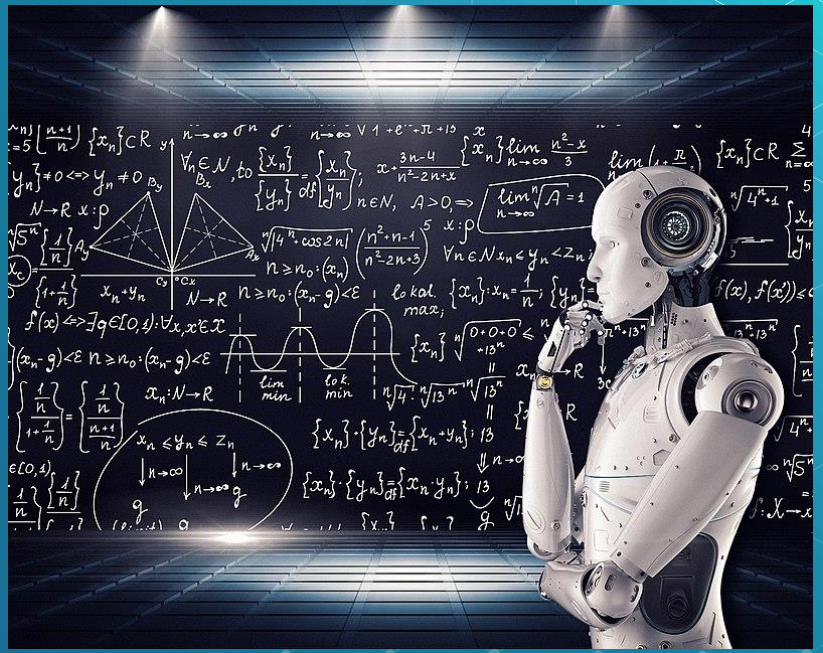
- 5. Year Computer Science
- Writing master's in Autonomous Driving
- Earlier been Project lead in 2 projects at Cogito NTNU (Emojify, CatMatch)

1 Introduction

Welcome to deep learning

What is AI?

- Defining AI is hard
  - John McCarthy:
    - "It is the science and engineering of making intelligent machines, especially intelligent computer programs."
  - What is intelligence?
    - Doing the correct thing or something like that



Quote source: http://jmc.stanford.edu/artificial-intelligence/what-is-ai/index.html
Image source: https://upload.wikimedia.org/wikipedia/commons/thumb/9/9d/Artificial_intelligence_Reasoning%2C_problem-solving.jpg/800px-Artificial_intelligence_Reasoning%2C_problem-solving.jpg

What is Deep Learning?

If statement

Linear regression

Neural networks

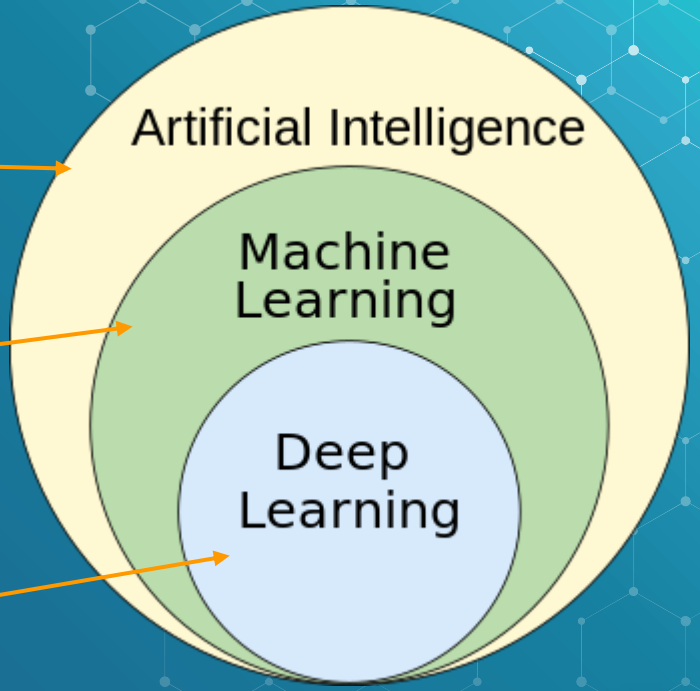


Image source:
https://upload.wikimedia.org/wikipedia/commons/1/1b/AI_hierarchy.svg

What is a Neural Network?

- Often use Artificial Neural Network (ANN)
- Consists of 3 parts
  - Input layer
  - Hidden layers
  - Output layer
- Each layer consists of artificial neurons
  - $a(\mathbf{x}) = f(\mathbf{w} \cdot \mathbf{x} + \mathbf{b})$
  - x - input
  - w - learned weights
  - b - learned bias
  - f - activation function (chosen beforehand)

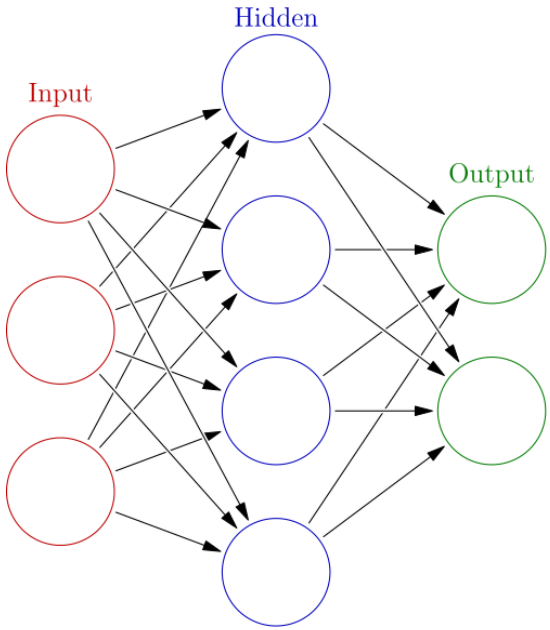


Image source: https://upload.wikimedia.org/wikipedia/commons/thumb/4/46/Colored_neural_network.svg/1200px-Colored_neural_network.svg.png

Training a neural network

- 3 main methods
  - **Supervised Learning**
    - We know the correct output
  - **Unsupervised learning**
    - We have no correct output
  - **Reinforcement learning**
    - Give the model a reward based on how correct it is
- *We will only care about **supervised learning*** today

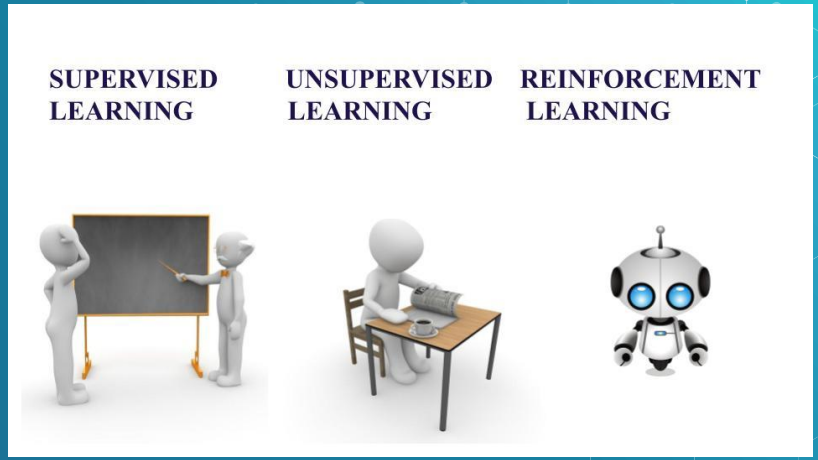


Image source: https://www.aitude.com/wp-content/uploads/2020/01/Untitled-presentation-1-1.jpg

What is a Neural Network? (2)

Steps for training a ANN with supervised learning

$$a(\mathbf{x}) = f(\mathbf{w} \cdot \mathbf{x} + \mathbf{b})$$

1. Randomly initialize the weights **w** and bias **b**.
2. Do a *forward pass* through the network (calculate the result for an input **x**). Thus obtain the predicted result $\hat{\mathbf{y}}$.
3. Using the loss function **L**, calculate $\mathbf{L}(\hat{\mathbf{y}}, \mathbf{y})$
4. Do backpropagation to find the gradient in the network
5. Use the optimizer, like SGD, to use the gradient to optimize the weights
6. Repeat from 2. until satisfied

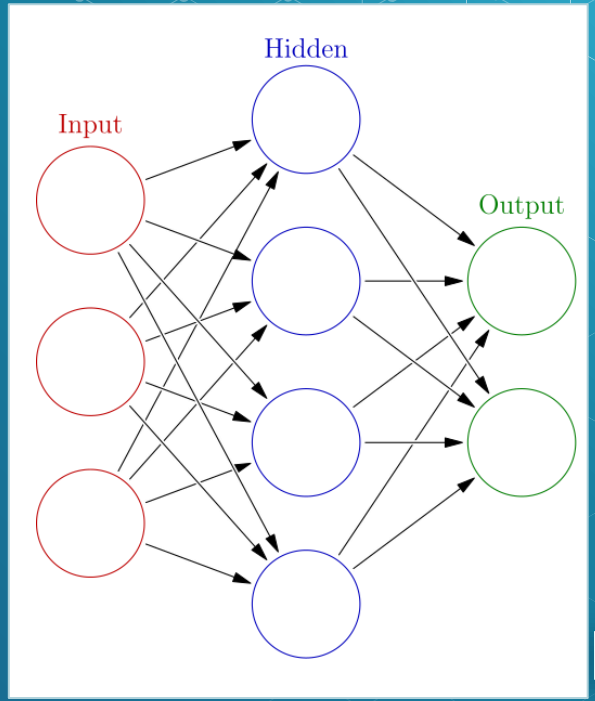


Image source: https://upload.wikimedia.org/wikipedia/commons/thumb/4/46/Colored_neural_network.svg/1200px-Colored_neural_network.svg.png

Architecture of a Neural Network

Architecture of VGG16
a CNN

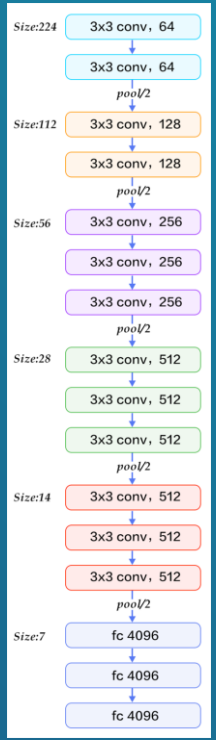


Image source: https://www.kaggle.com/code/blurredmachine/vggnet-16-architecture-a-complete-guide

What is a CNN?

- It is a method to analyze images in a neural network
- Groups parts of the image to produce a smaller representation called a **feature map**

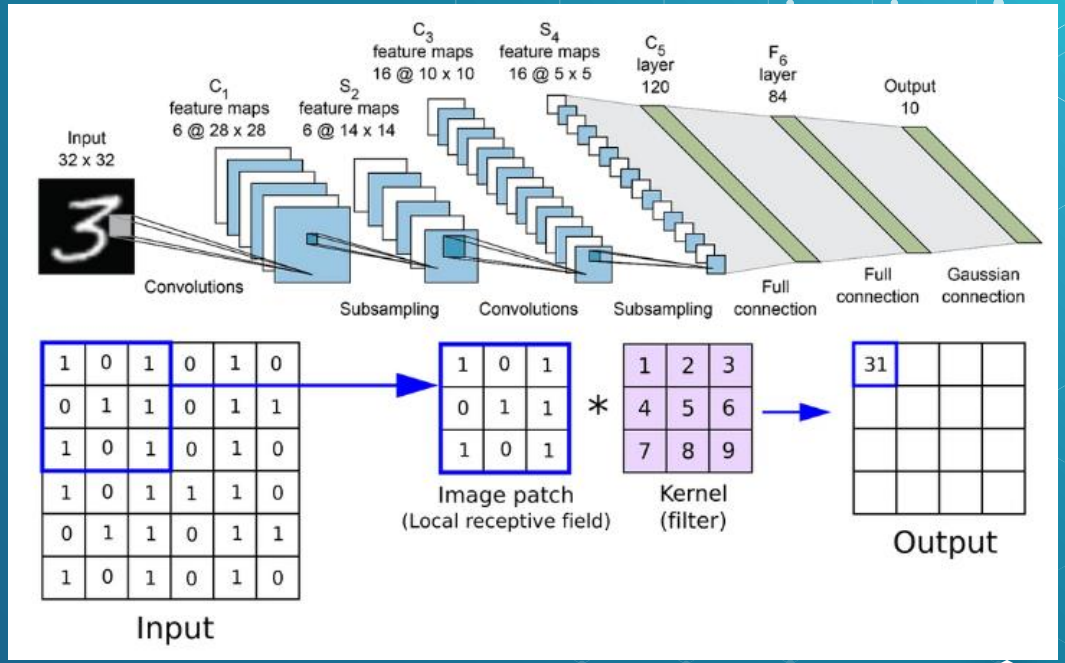
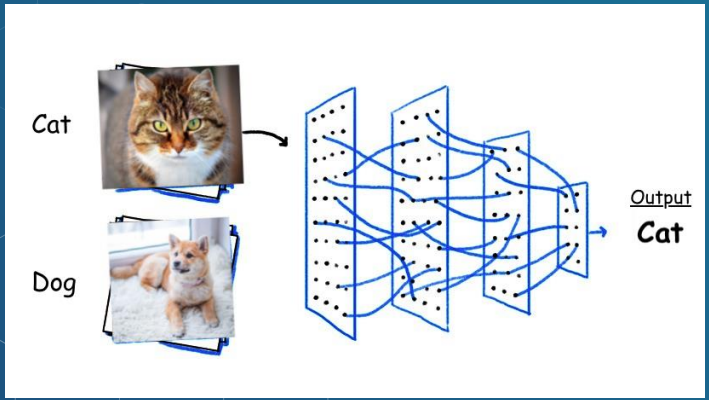


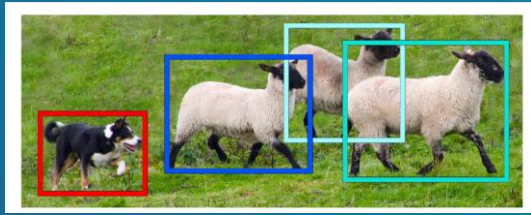
Image source: https://assets-global.website-files.com/614c82ed388d53640613982e/646371e3bdc5ca90dee5331b_convolutional-neural-network%20(1).webp

What can a CNN do? - examples

Classification



Object Detection



Semantic Segmenation

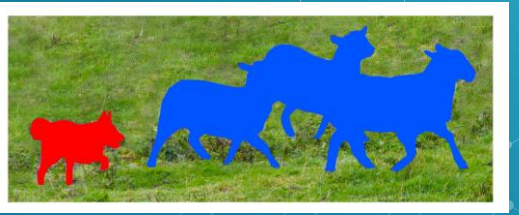


Image sources:
Left: https://towardsdatascience.com/10-papers-you-should-read-to-understand-image-classification-in-the-deep-learning-era-4b9d792f45a7
Middle and right: https://manipulation.csail.mit.edu/segmentation.html

Datasets

- Is a set of data that is used to train the model
- Especially used in supervised learning to ensure having the "correct" values - *ground truth*

ImageNet (14 mill labeled images)

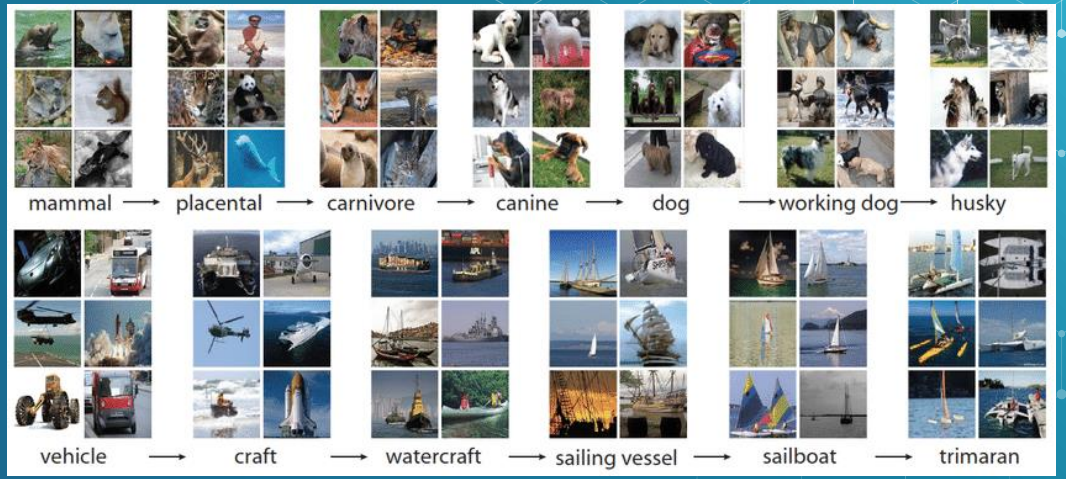


Image source: https://blog.roboflow.com/introduction-to-imagenet/

Typical datasets

- Premade datasets
  (ImageNet, CIFAR100, Cityscapes, nuScenes...)
- CSV files
- Find open datasets on https://kaggle.com

Important aspects of datasets

- Balanced distribution of data
- Include all relevant situations
- Large (usually several 1000 elements)
- No annotation errors

Processing of dataset during training

- The dataset is split into
  - Training dataset (often 80%)
  - (Validation dataset) (often 10%)
  - Testing dataset (10% - 20%)
- Processed in batches
- Format the data to a way the model efficiently understands

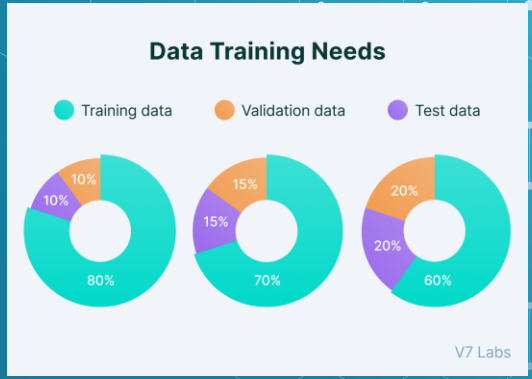


Image source: https://www.v7labs.com/blog/train-validation-test-set

PyTorch

- Is research industry standard library for deep learning
- Includes all you tools you need to build, train and run a deep learning model
- Also includes some premade and pretrained models in torchvision
- Alternatives: Tensorflow, JAX

Image source: https://github.com/pytorch/pytorch/blob/main/docs/source/_static/img/pytorch-logo-dark.png

2 Datasets

What the model is learning from

HOW TO REPRESENT DATA?

How to represent data?

- Images
  - Each pixel is a vector of their red, green and blue value [255, 165, 0]
  - Thus a image that is 512 * 256 px has shape (512, 256, 3)
- Labels
  - Numbers
  - If your labels are [“cat”, “dog”, “mouse”]
  - The labels are cat=1, dog=2, mouse=3 and are represented by position in an array in the network

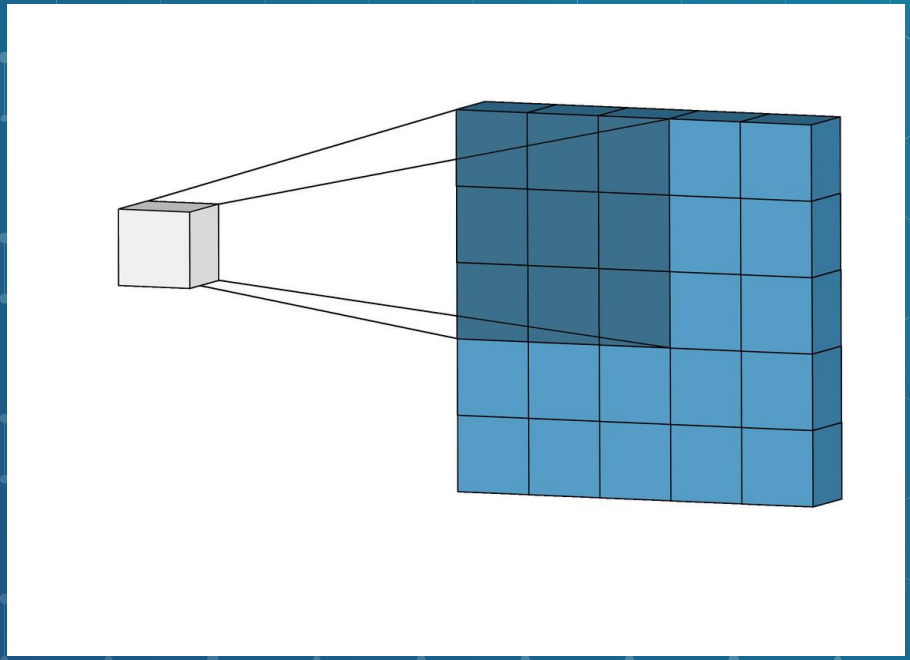
Datasets in PyTorch

- On to the Google Colab Notebook

3

CNNs

Convolutional Neural Networks



CONVOLUTION

Image source: https://medium.com/@nikitamalviya/convolution-pooling-f8e797898cf9

Convolution - the continuous math def

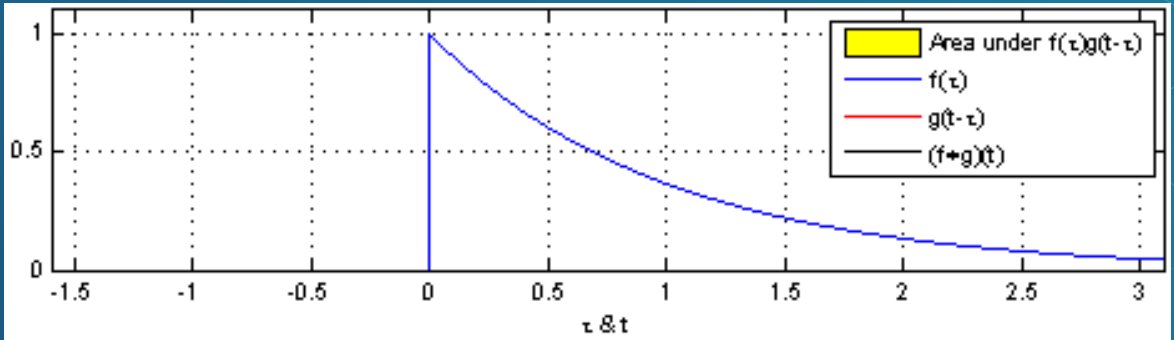
$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau.$$

Source: en.wikipedia.org/wiki/Convolution

Convolution - mathematical definition

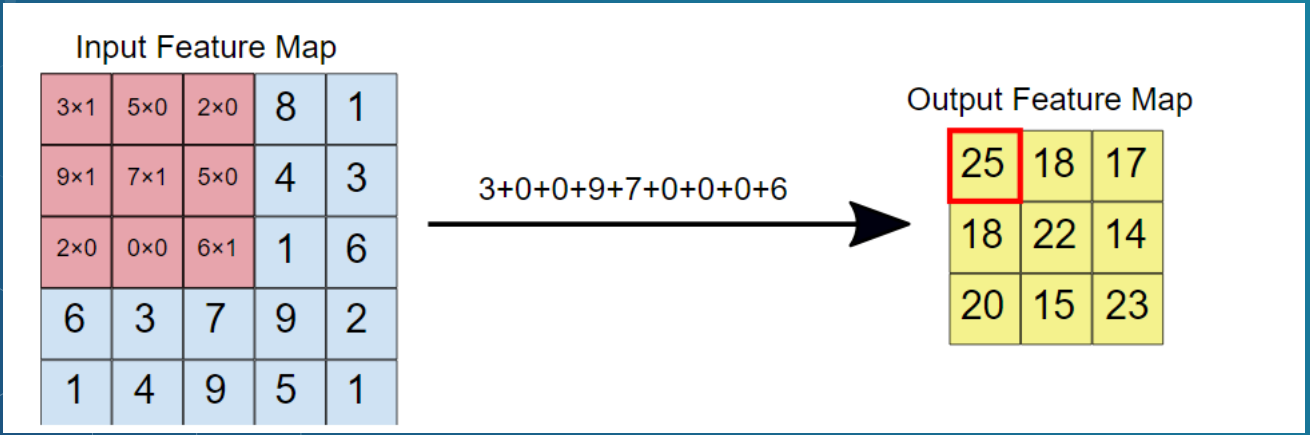
A way to combine two functions

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau.$$



Source: en.wikipedia.org/wiki/Convolution

Convolution - on images



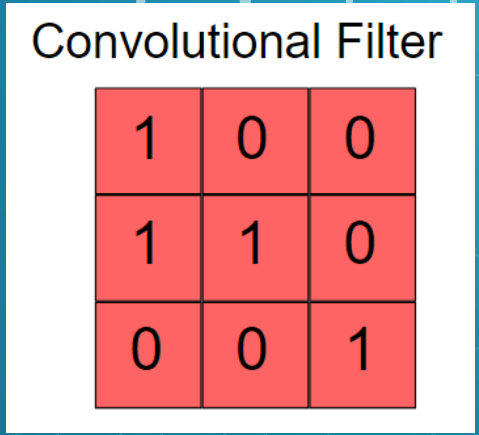


Image source: https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks

Convolution - on images

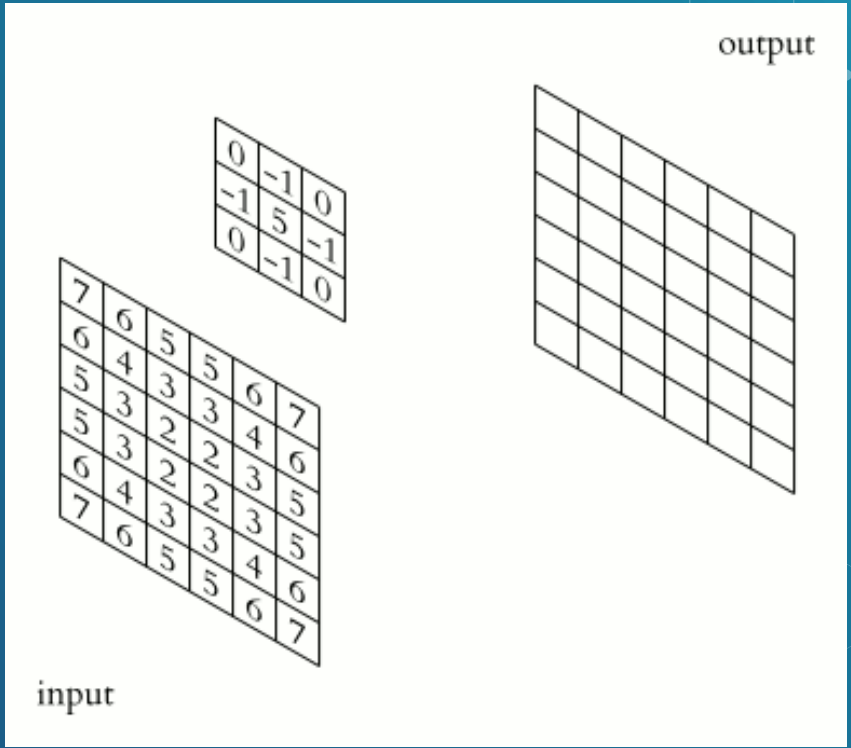


Image source:
https://upload.wikimedia.org/wikipedia/commons/thumb/1/19/2D_Convolution_Animation.gif/330px-2D_Convolution_Animation.gif

Exercise!

- Implement a convolution on an image!

Convolutional Layers

- We are trying to find the kernel!
- The weights in a convolutional layer denote the numbers that make up the kernel

Convolutional Neural Networks

- A CNN is an ANN with at least one convolutional layer
- Other layers:
  - Batch Normalization (BatchNorm)
  - Max Pool
- A CNN outputs a feature map
  - A lower-dimensionality representation of the image

Max Pooling

- Uses a n*n kernel and a stride
- Goes through the matrix like convolution,
- But chooses the maximum value instead

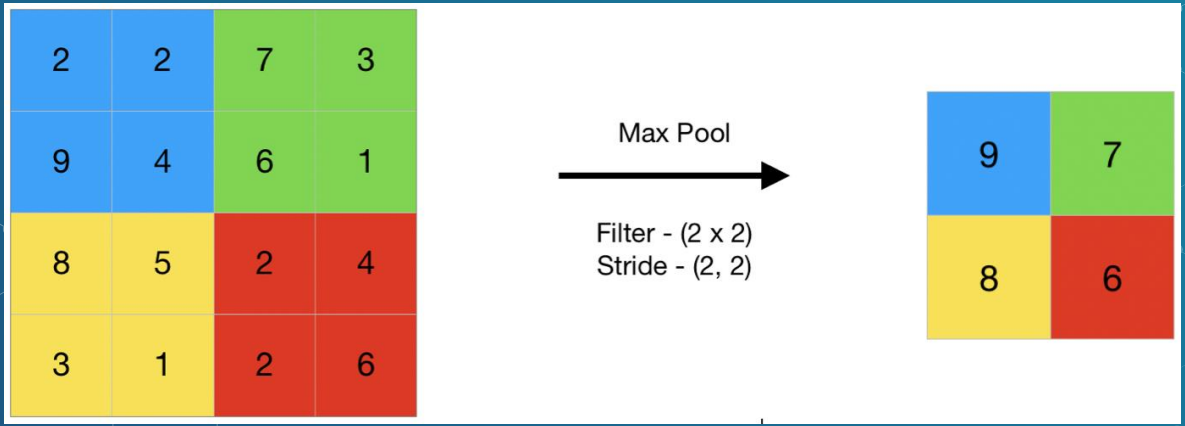


Image source: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

Max Pooling - Task!

- Let’s implement Max Pooling!

Batch Normalization

- Uses re-centering and scaling to make ANNs faster and more stable
- It normalizes the distribution of output values from the neurons
- This way large values don't impact the result that much

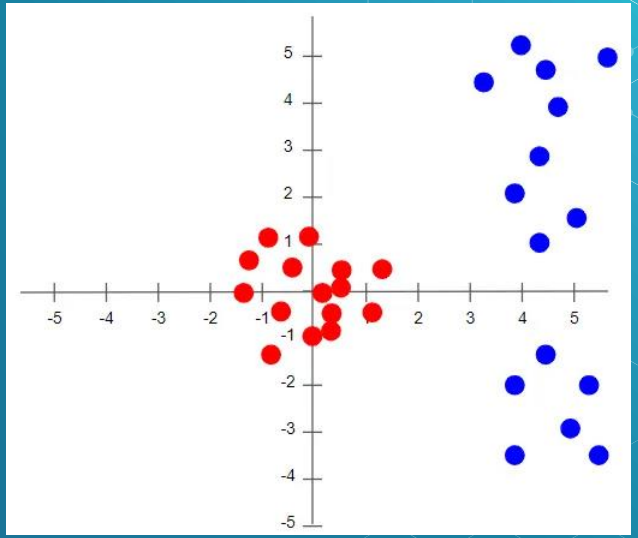


Image source: https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

4

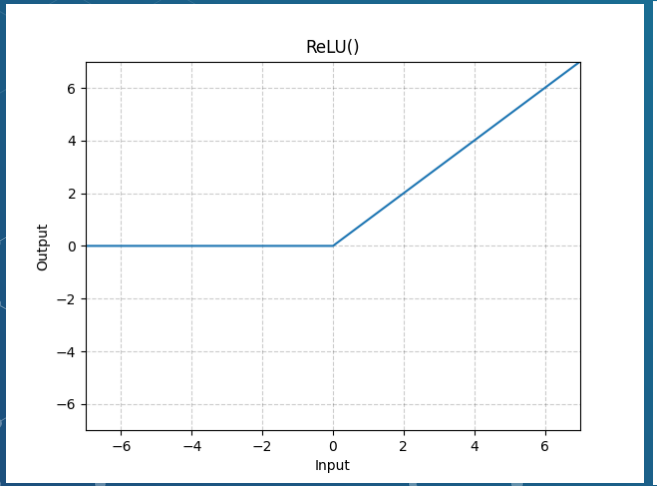
Training Loop

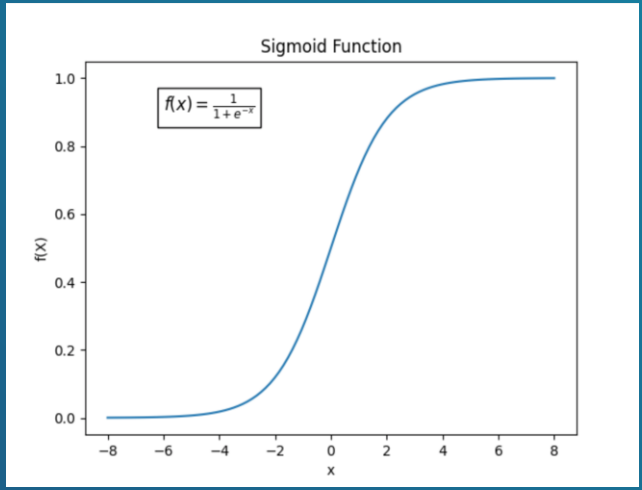
The most successful ones

Neural Networks - Activation Functions

- Introduces non-linearity into the network

$$a(\mathbf{x}) = f(\mathbf{w} \cdot \mathbf{x} + \mathbf{b})$$





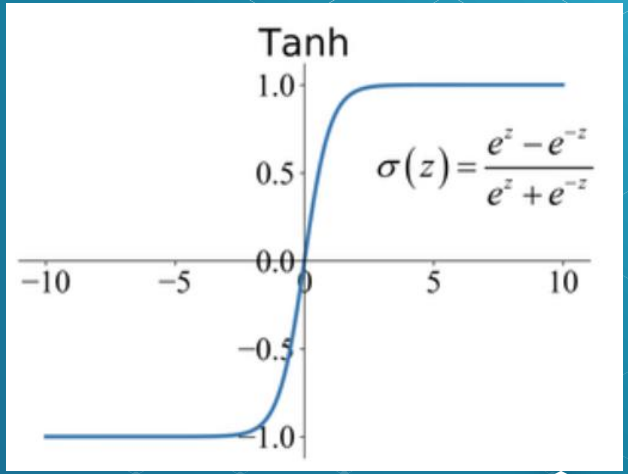


Image source:
Left: https://pytorch.org/docs/stable/_images/ReLU.png
Middle: https://raw.githubusercontent.com/Codecademy/docs/main/media/sigmoid-function.png
Right: https://production-media.paperswithcode.com/methods/Screen_Shot_2020-05-27_at_4.23.22_PM_dcuMBJl.png

Neural Networks - Loss

- Are function that calculate how “wrong” the prediction is.
- Their derivative decide the gradient
- Examples (for regression):

Steps for training a ANN with supervised learning

1. Randomly initialize **w** and **b**.
2. Do a *forward pass* to obtain **ŷ**.
3. Calculate loss: **L**(**ŷ**, **y**)
4. Do backpropagation
5. Use the optimizer (e.g.) SGD to improve weights.
6. Repeat from 2. until satisfied

Mean Squared Error

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2.$$

Mean Absolute Error

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|\, y_i - \hat{y}_i \,\right|$$

Image sources:
Left: en.wikipedia.org/wiki/Mean_squared_error
Right: https://editor.analyticsvidhya.com/uploads/42439Screenshot%202021-10-26%20at%209.34.08%20PM.png

Neural Networks - Loss (2)

- For classification

Cross Entropy Loss

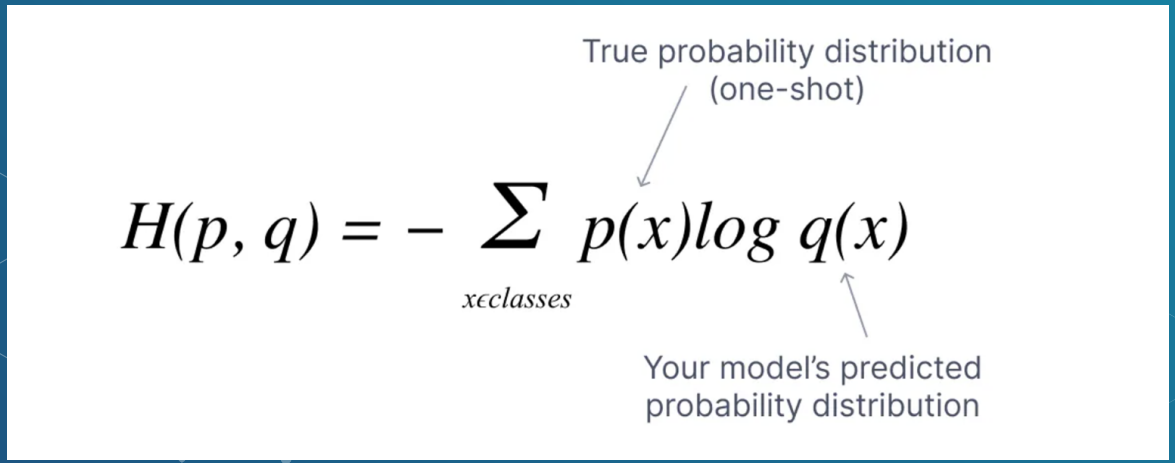


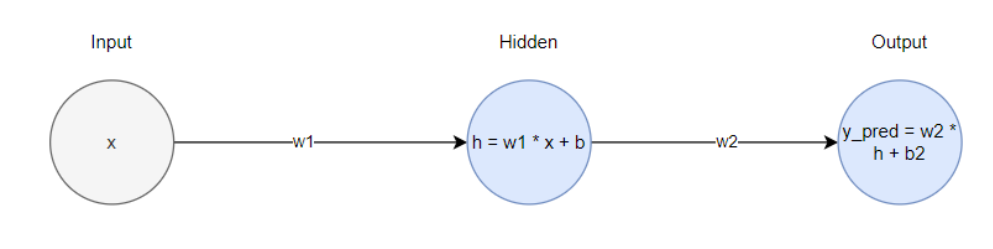
Image source: https://www.v7labs.com/blog/cross-entropy-loss-guide

Steps for training a ANN with supervised learning

1. Randomly initialize **w** and **b**.
2. Do a *forward pass* to obtain $\hat{\mathbf{y}}$.
3. Calculate loss: $\mathbf{L}(\hat{\mathbf{y}}, \mathbf{y})$
4. Do backpropagation
5. Use the optimizer (e.g.) SGD to improve weights.
6. Repeat from 2. until satisfied

Neural Networks - Backpropagation

- Is an algorithm to compute the gradient on each network node with respect to the weights
- (Using MSE as loss function)



Steps for training a ANN with supervised learning

1. Randomly initialize **w** and **b**.
2. Do a *forward pass* to obtain **ŷ**.
3. Calculate loss: **L**(**ŷ**, **y**)
4. Do backpropagation
5. Use the optimizer (e.g.) SGD to improve weights.
6. Repeat from 2. until satisfied

1. **Output Layer Gradients**:
   - Gradient of Loss w.r.t w_2: $\frac{\partial \text{Loss}}{\partial w_2} = (y_{\text{pred}} - y_{\text{true}}) \cdot h$
   - Gradient of Loss w.r.t b_2: $\frac{\partial \text{Loss}}{\partial b_2} = (y_{\text{pred}} - y_{\text{true}})$
2. **Hidden Layer Gradients**:
   - Gradient of Loss w.r.t w_1: $\frac{\partial \text{Loss}}{\partial w_1} = (y_{\text{pred}} - y_{\text{true}}) \cdot w_2 \cdot x$
   - Gradient of Loss w.r.t b_1: $\frac{\partial \text{Loss}}{\partial b_1} = (y_{\text{pred}} - y_{\text{true}}) \cdot w_2$

Neural Networks - Optimizers

- Is the algorithm to update the weights given the gradient from backpropagation
- Cost = Average Loss for entire dataset
- Formula:

w_new = w - learning_rate * gradient

(learning_rate is a pre-decided hyperparamter)

Steps for training a ANN with supervised learning

1. Randomly initialize **w** and **b**.
2. Do a *forward pass* to obtain **ŷ**.
3. Calculate loss: **L**(**ŷ**, **y**)
4. Do backpropagation
5. Use the optimizer (e.g.) SGD to improve weights.
6. Repeat from 2. until satisfied

Gradient Descent

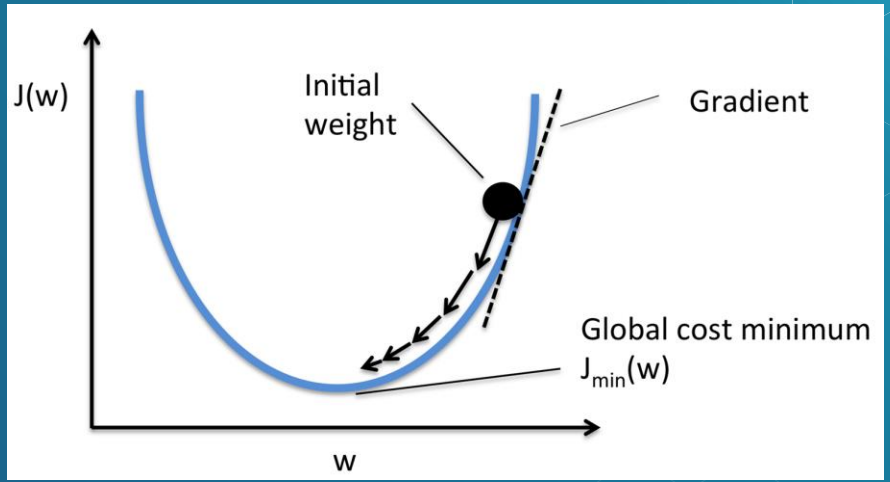


Image source: https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/

Neural Networks - Optimizers (2)

- Usually Stochastic Gradient Descent is used
- Instead of using the average gradient for the entire dataset, uses the gradient from a small subset of the dataset

Steps for training a ANN with supervised learning

1. Randomly initialize **w** and **b**.
2. Do a *forward pass* to obtain $\hat{\mathbf{y}}$.
3. Calculate loss: $\mathbf{L}(\hat{\mathbf{y}}, \mathbf{y})$
4. Do backpropagation
5. Use the optimizer (e.g.) SGD to improve weights.
6. Repeat from 2. until satisfied

```python
import numpy as np

# Gradient Descent
def gradient_descent(x, y, theta, alpha, num_iters):
    m = len(y)
    for i in range(num_iters):
        h = np.dot(x, theta)
        loss = h - y
        gradient = np.dot(x.T, loss) / m
        theta = theta - alpha * gradient
    return theta

# Stochastic Gradient Descent
def stochastic_gradient_descent(x, y, theta, alpha, num_iters):
    m = len(y)
    for i in range(num_iters):
        for j in range(m):
            h = np.dot(x[j], theta)
            loss = h - y[j]
            gradient = x[j].T * loss
            theta = theta - alpha * gradient
    return theta
```

Code source: https://medium.com/@dhirendrachoudhary_96193/what-is-the-difference-between-stochastic-gradient-descent-sgd-and-gradient-descent-gd-10e7d8019018

Neural Networks - Optimizers (3)

- Other Optimizers
  - Adam
  - AdamW

- Converge faster than SGD by using
  - Adaptive learning rate
  - AdamW has weight decay (to discourage large weights)

Steps for training a ANN with supervised learning

1. Randomly initialize **w** and **b**.
2. Do a *forward pass* to obtain $\hat{\mathbf{y}}$.
3. Calculate loss: $\mathbf{L}(\hat{\mathbf{y}}, \mathbf{y})$
4. Do backpropagation
5. Use the optimizer (e.g.) SGD to improve weights.
6. Repeat from 2. until satisfied

CNN training loop in PyTorch

- Let’s see some code

5

Specific CNN Architectures

The most successful ones

VGG

- Inputs 224 * 224 pixel RGB images
  - For ImageNet they cropped out the center 224*224 patch of the image
- Composed of 16 layers
  - 13 Convolutional 3 x 3 layers
  - Ends with 3 fully connected layers
  - Uses ReLU as the activation function

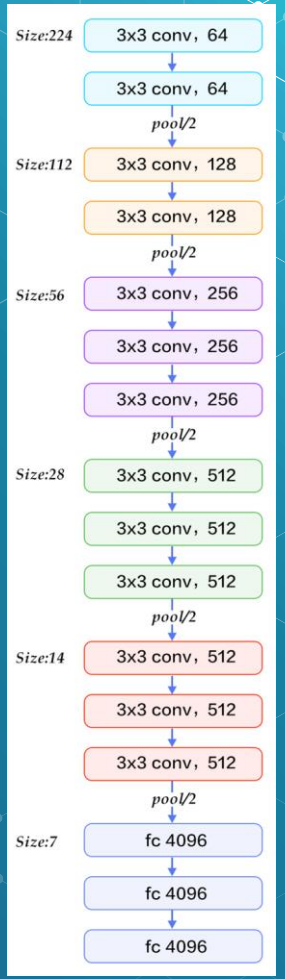


Image source: https://www.kaggle.com/code/blurredmachine/vggnet-16-architecture-a-complete-guide

ResNet

- Introduces the residual connection
  - Skips layers and adds a previous input to the current output.
- Solves the vanishing gradients problem of deep neural networks

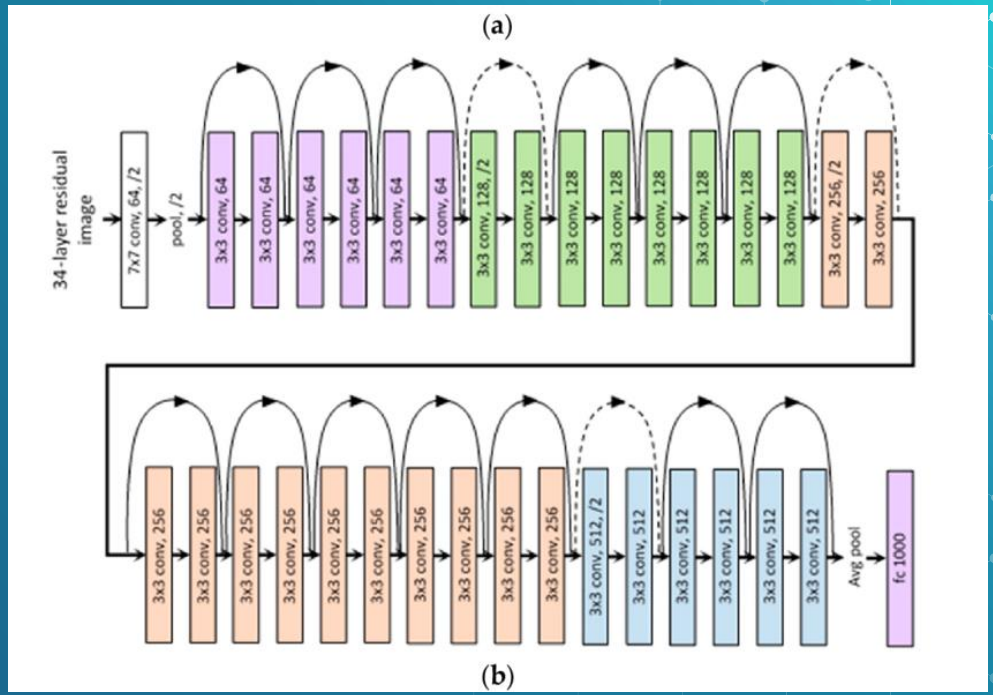


Image source: https://medium.com/@siddheshb008/resnet-architecture-explained-47309ea9283d
ResNet paper: https://arxiv.org/pdf/1512.03385.pdf

EfficientNet

- A CNN that builds upon MobileNetV2 and uses MBConv layers
- Introduces an optimal scaling of depth (number of layers) and width (number of neurons per layer) and image resolution by accuracy and desired parameters
- The most efficient CNN if you want to use one

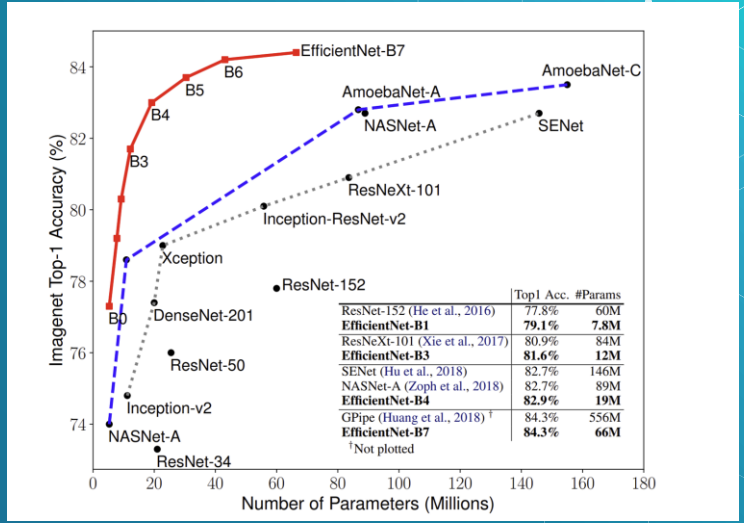
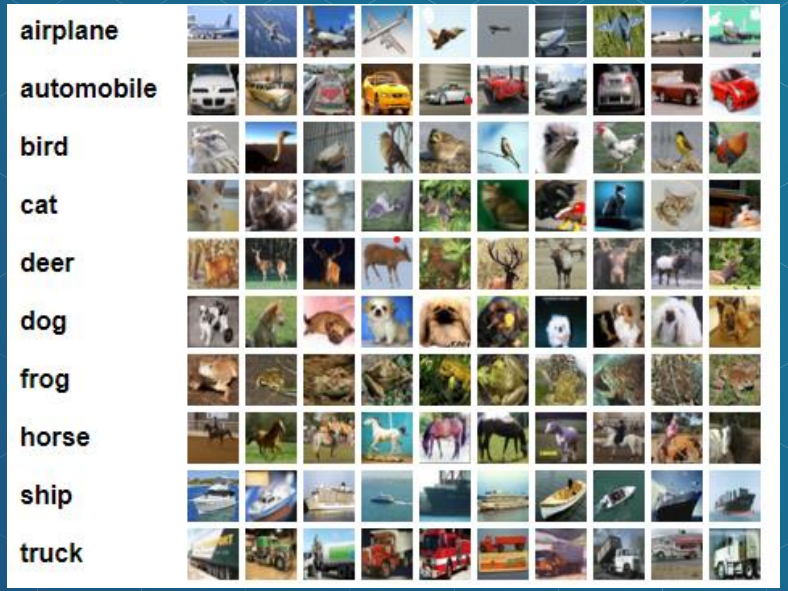


Image source and EfficientNet paper: arxiv.org/pdf/1905.11946.pdf

CIFAR10



COMPETITION!

Image source: https://www.cs.toronto.edu/~kriz/cifar.html

Competition: Accuracy on CIFAR10

- **Goal**: Train and validate a custom CNN on the CIFAR10 dataset

Rules

1. You cannot use pretrained or prebuilt networks, like ResNet50 from torchvision
2. Use the last 20% of the dataset for testing. This data cannot be trained on. (See `cifar_test` and `cifar_trainable`)
3. You can only train on the trainable CIFAR10 data, and no other data.
4. You can create as big network as you choose using anything from torch.nn, as long as you build it yourself.
5. You have to complete the challenge within the time limit.