MIDDLE EAST TECHNICAL UNIVERSITY

DI504 Foundations of Deep Learning

Summary & Conclusions

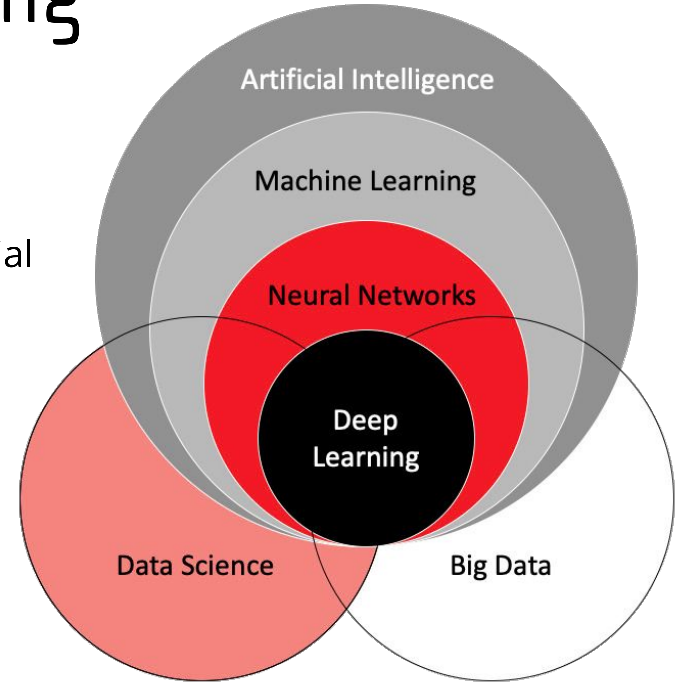
We learned about "deep learning"

Artificial Intelligence: is human intelligence exhibited by machines.

Machine Learning is an approach to achieve artificial intelligence

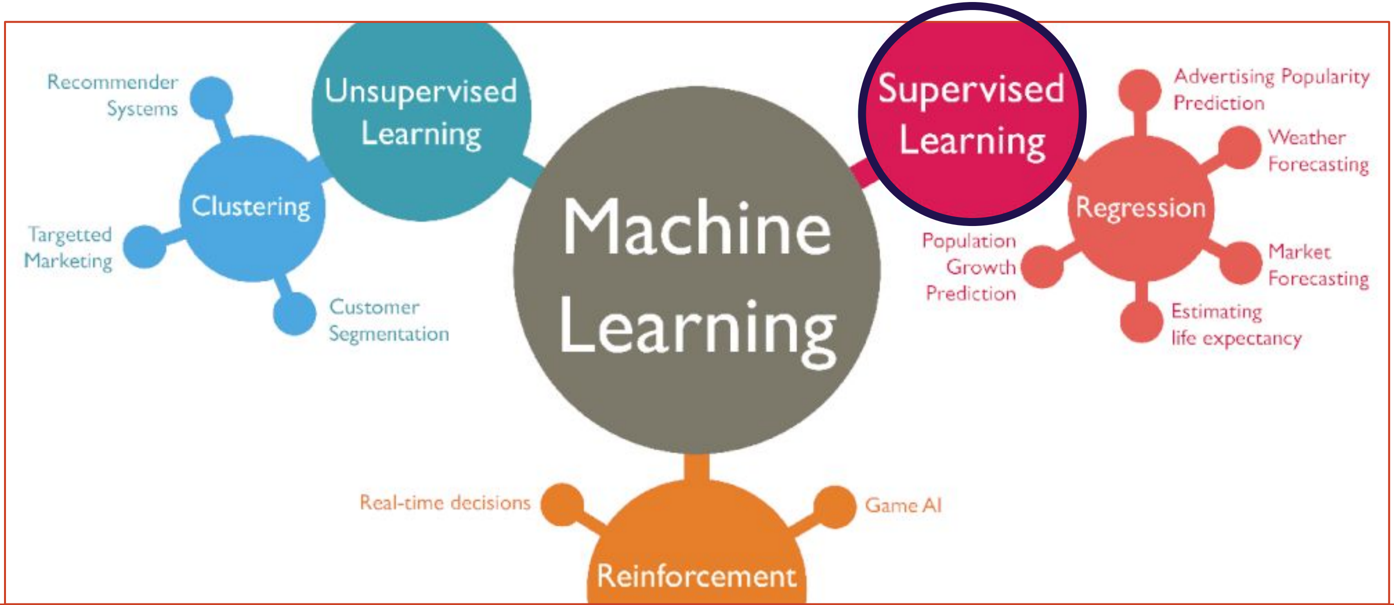
Neural Networks (or Nets) is a one of the techniques for implementing machine learning.

Deep Learning is a subfamily of machine learning methods, based on "deep" (many layered) neural networks.



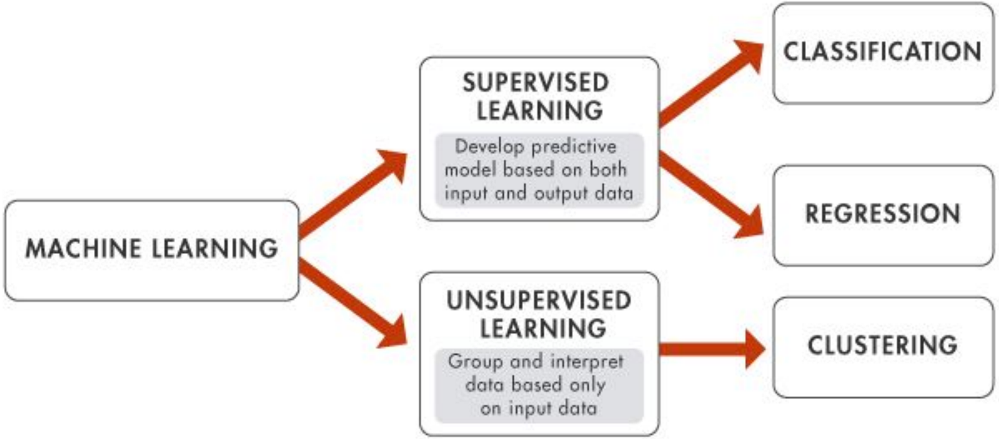
We worked on "Supervised Learning"

- ML focuses on the development of computer programs that can access data and use it to learn for themselves, **using the data**.
- ML algorithms are often categorised according to how they are supervised.



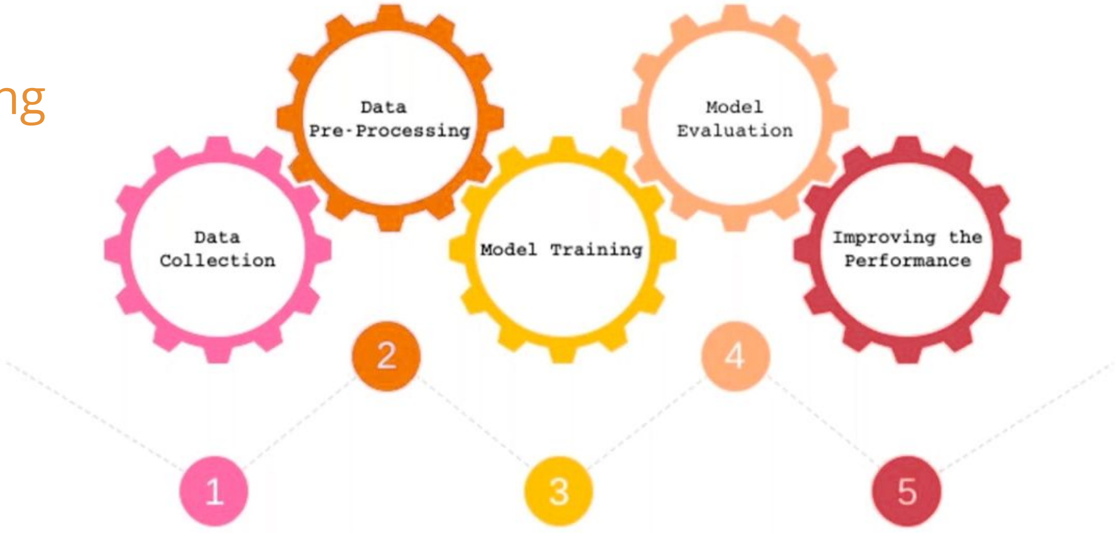
Supervision in ML

- Supervised learning (SL) is the machine learning task of learning a function that maps an input to an output based on example input-output pairs
- Unsupervised learning (UL) is a type of algorithm that learns patterns from untagged data. The hope is that, through mimicry, the machine is forced to build a compact internal representation of its world and then generate imaginative content.



We studied ML Workflow

1. Data Collection
2. Pre-processing / Feature Extraction
3. Model Training
4. Model Evaluation / Testing
5. Fine-Tuning / Parameter Optimization



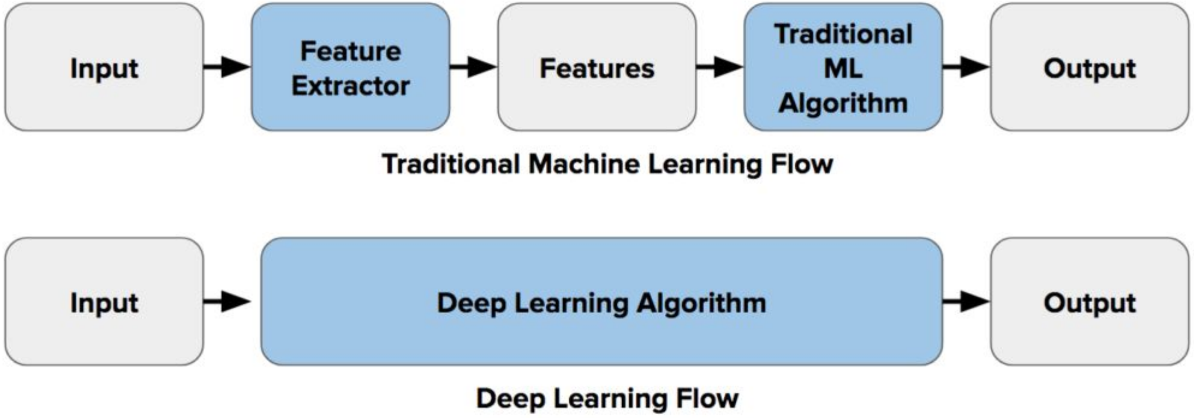
We learned about "Features"

- In ML a "***feature***" is an individual measurable property or characteristic of a phenomenon, that represents "something" (tangible or not) in the data.
- Features are higher level representations of "raw data".

- Before deep learning, in 2000s, the AI society of today focused mainly on "feature engineering", i.e. finding the right hand-crafted representation for the data that would make our ML model perform the best.

We learned about “Deep vs ML Features”

- [Wikipedia] Deep learning (DL) is a class of ML algorithms that uses multiple layers to “<u>progressively extract higher-level features from the raw input</u>”.
- DL algorithms learn their own features. Hence, they do not require an additional feature extraction step, like traditional ML algorithms do.



We learned about “cross-validation”

You can:

1. Split data into train, val, and test; choose hyperparameters on validation and evaluate on test sets. Decide the hyper-parameters on validation data. Finally test your success on Test data
   - Yeah, that’s the way to do it!

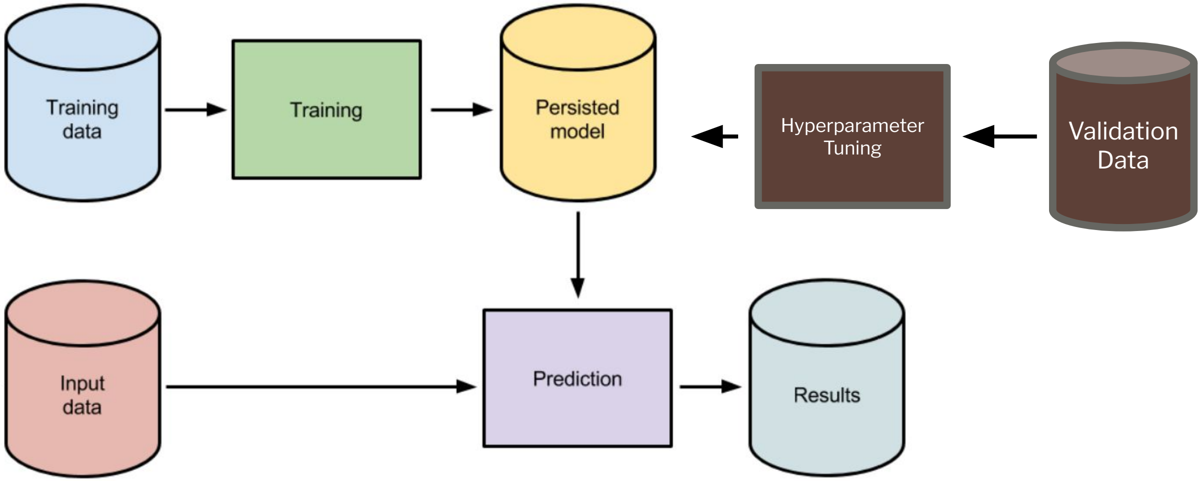
Training set	Validation set	Test set

- This is called **cross-validation**. It is a technique to assess how the results of a statistical analysis (such as a machine learning problem) will generalize to an independent data set.

METU

We learned about “ML Pipeline”

- *The simplest machine learning pipeline looks like this*



We learned about "Problems in Deep Learning"

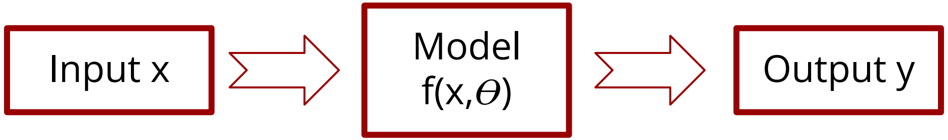
- **Detection**: Designate the existence of a «type of signal» (i.e. Object) in a given signal.
- **Classification/Recognition**: Given a signal with an object, find out what that object is. In other words, classify it in a class from a set of predefined categories.
- **Localization** : Find where the region-of-interest is, within a signal.
- **Segmentation**: Classify every signal sample (pixel, sound clip) in a signal to a class according to its context.
- **Regression**: Reconstruct a function with a given an input and an output signal.
- **Generation**: Create an original signal, given a dataset of similar signals.

We learned about the “Score Function”

Let’s remember what the score function was

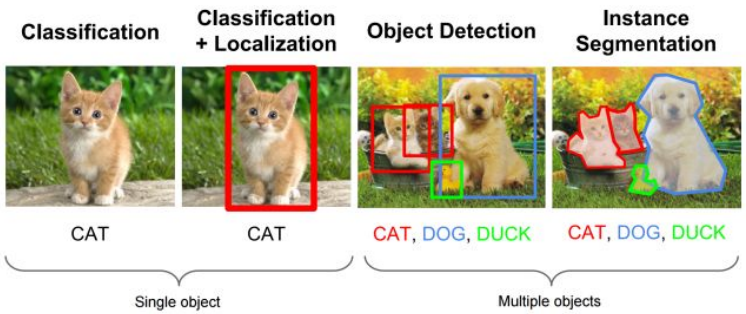
- $f(x,\Theta)$ is basically a function of input x and model parameters Θ, that output a score value y.

$$f(x,\Theta) = y$$



We learned about the “Ground Truth”

- A ground-truth dataset is a regular dataset, but with *annotations* added to it.
- *Annotations* can be boxes drawn over images, written text indicating samples, a new column of a spreadsheet or anything else the machine learning algorithm should learn to output.
- Depending on problem type, the character of annotations may differ:



We learned about the “Loss Function”

- So for this apple, how good is this score function

We define a loss function for

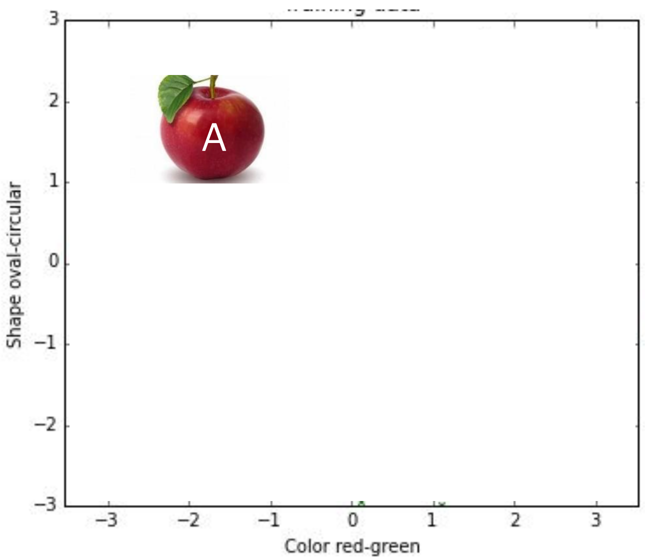
$L(x_i, g_i) = g[f(x_i,\theta), g_i]$
$L(x_i, g_i, \theta) = |f(x_i,\theta) - g_i| = +2.5.$

For example: 2D input (colour and shape) for a model that predicts how juicy an apple is

$f(x,\theta) = 2\cdot x_2 - 3\cdot x_1$

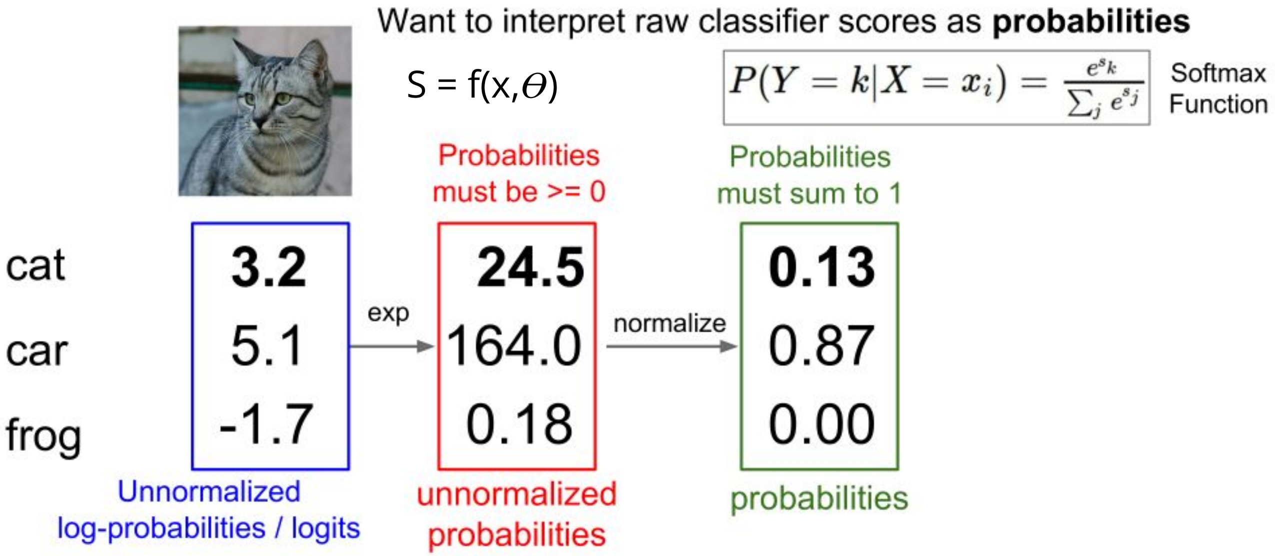
Say $x = [-1.5 +1.5] \rightarrow y = +7.5$
(but the ground truth says it is 5.0 juicy.)

Gourmets from all around the world decided on this value (5.0) for this apple.



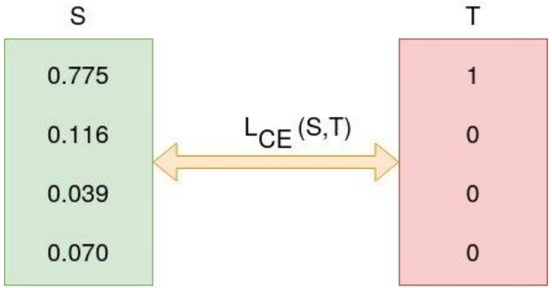
We learned about "Soft-max"

- Soft-Max is a score function normalizer for categorical scores.
- It converts real value scores to probabilities (that sum upto 1).



Cross-Entropy Distance (i.e. loss)

- Cross-entropy is commonly used in machine learning as a loss function.
- Cross-entropy is a measure from the field of information theory, building upon entropy and generally calculating the difference between two probability distributions.
- It is closely related to but is different from KL divergence that calculates the relative entropy between two probability distributions, whereas cross-entropy can be thought to calculate the total entropy between the distributions.

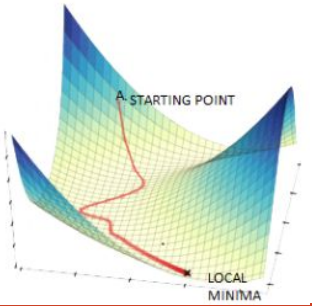


METU

We learned about "Optimization"

"By changing the model parameters (Θ) and "iteratively" checking if the score function is doing well (using the loss function), we find the correct parameters."

- Ok, how? Follow the slope (Gradient Descent)
  - For each iterative result, we may follow a direction, if the loss has a decreasing trend in that direction.
  -



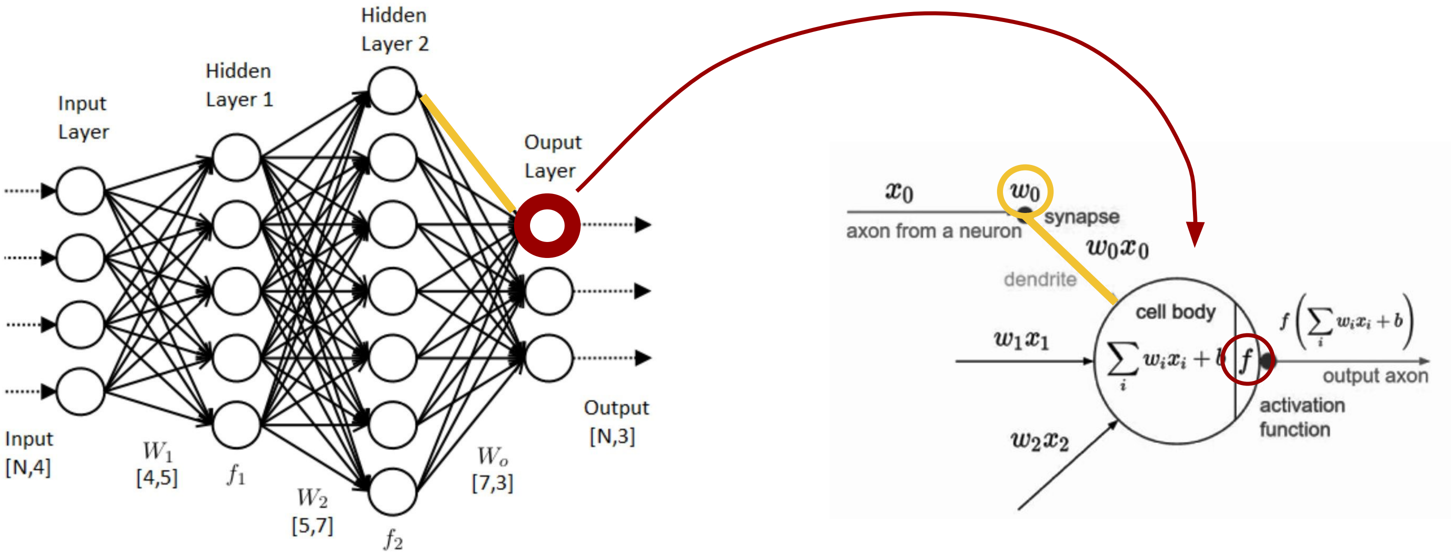
We learned about "Training"

Up to know we talked about

- Score function : $f(x_i,\Theta) = y_i$
- Ground Truth : g_i
- Loss function : $L(x_i , g_i) = g [f(x_i,\Theta) , g_i]$
- Gradient Descent : $$\Theta_{\text{new}} = \Theta_{\text{old}} - \lambda \frac{\delta L (x_i, g_i, \Theta_{\text{old}})}{\delta\Theta_{\text{old}}}$$

We learned about "Artificial Neural Networks"

What defines an ANN is the total set of **"Weights"** and each connection in an ANN represents a weight.



We learned about "Computational Graphs"

- A computational graph is a way to represent a mathematical function in the language of graph theory.
  - Graph Theory in a nutshell: *nodes are connected by edges, and everything in the graph is either a node or an edge.*
  - Simple Example:

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

We learned about “Backpropagation”

- Backpropagation (BP) is short for "backward propagation of errors," is an algorithm for supervised learning of artificial neural networks using gradient descent.
- Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the ANN’s weights.

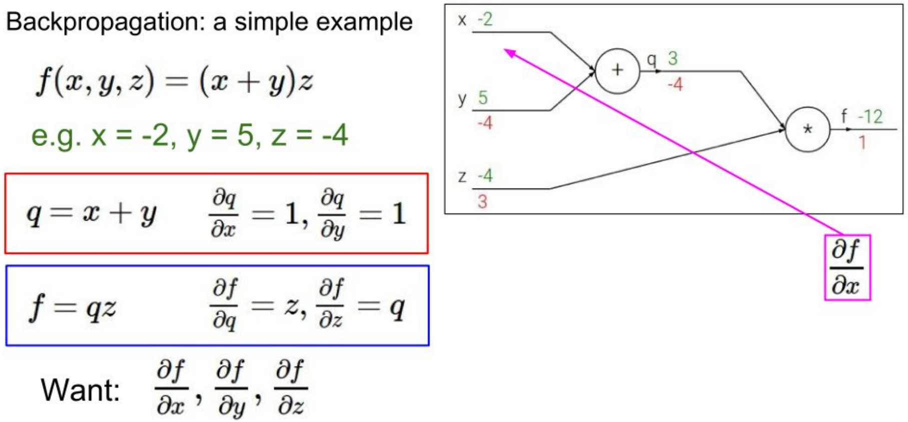
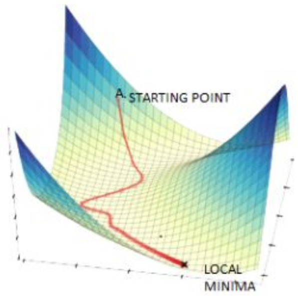
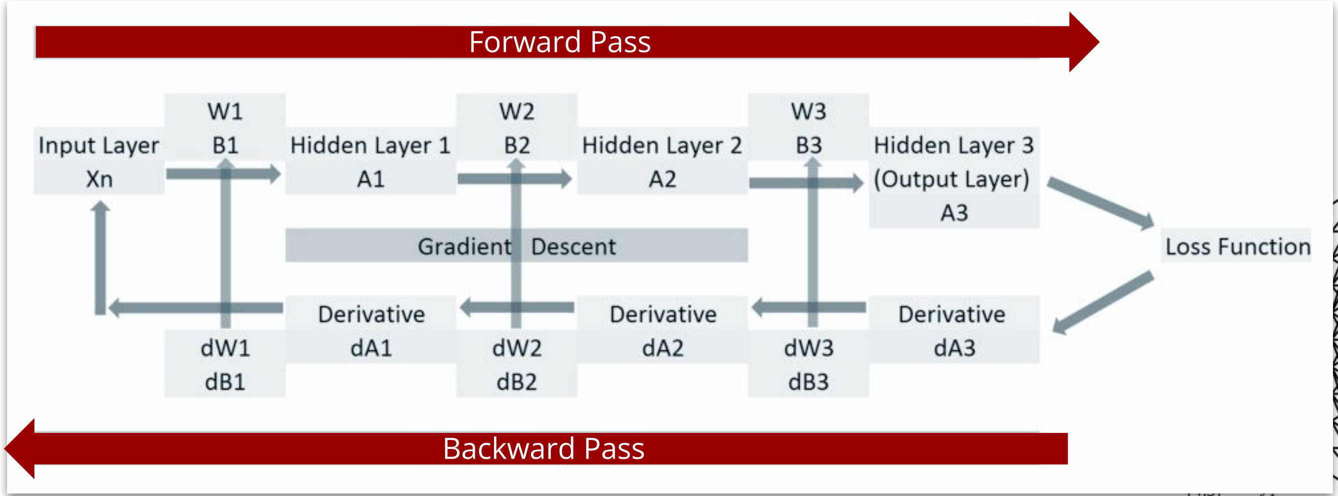


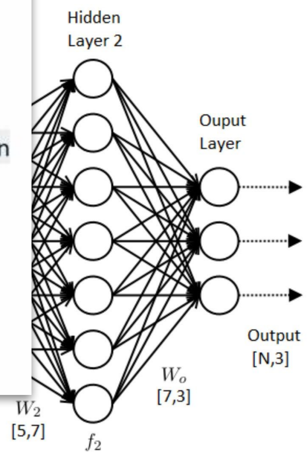
Figure from here.



We learned about "Training Neural Nets"

- Training consists of two main steps:
  - Forward pass: Function Evaluation,
  - Backward bass: Backpropagation.





We learned about “Convolution”

- In image processing, because the signal and the kernel are two-dimensional, the summation operator becomes a double summation.
- In this sense, the result of the convolution is the summation of the pixel-wise multiplication of the kernel values, with the neighbouring pixel values, centred around an arbitrary pixel

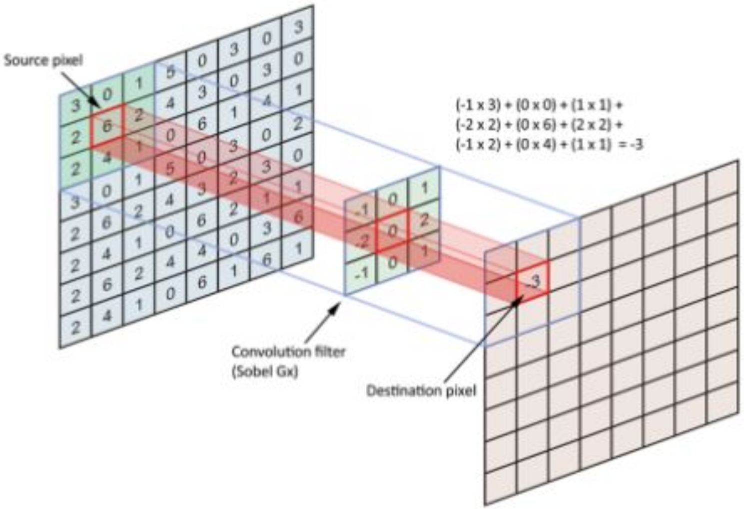
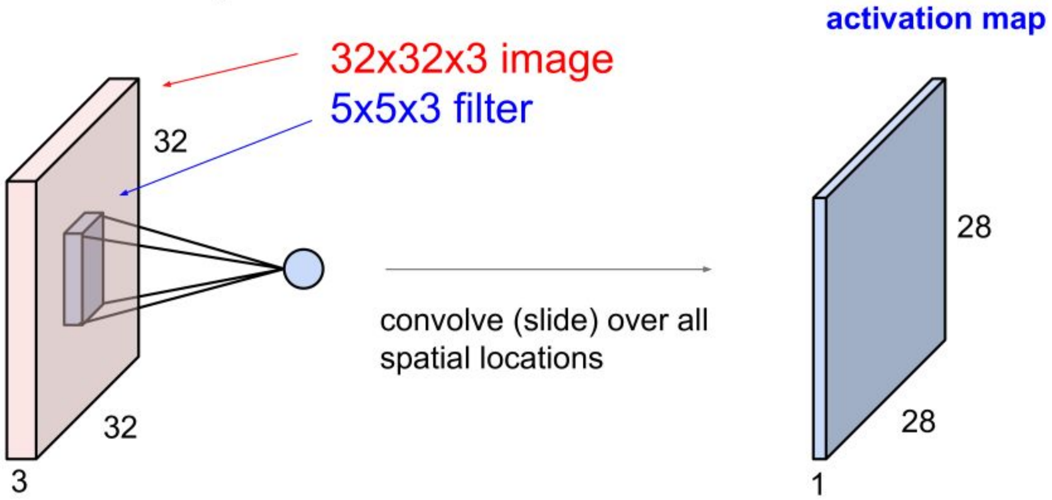


Figure 2. Convolution is the summation of the pixel-wise multiplication of the kernel values, with the neighbouring pixel values, centred around an arbitrary pixel.

We learned about “Convolutional Layers”

- When you “convolve” the filter all over the spatial dimensions, each sum creates an “activation”.

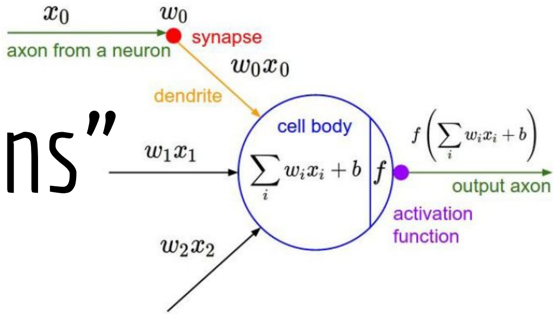


We learned about other "Layers Types"

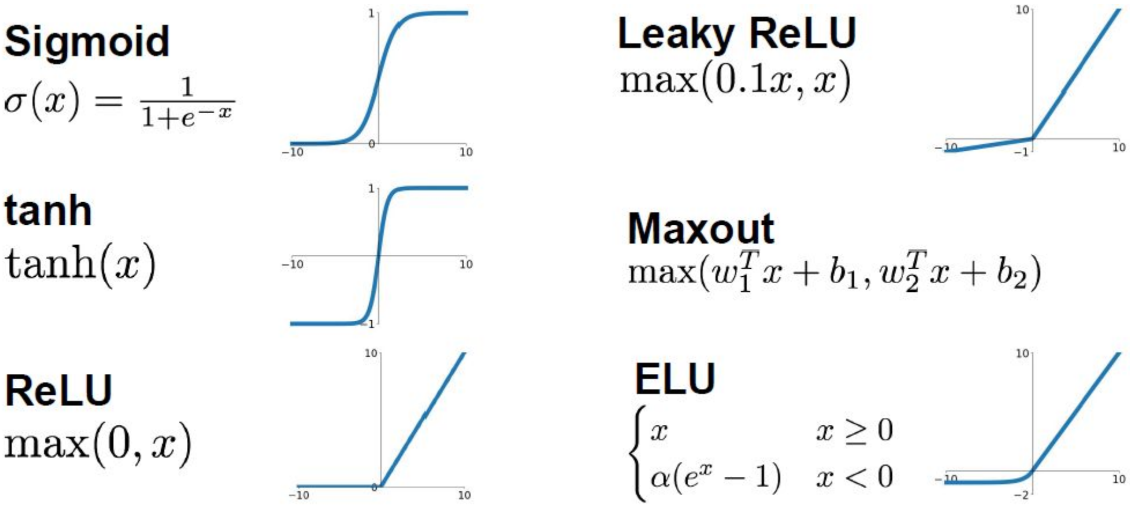
Let's categorize layers first:

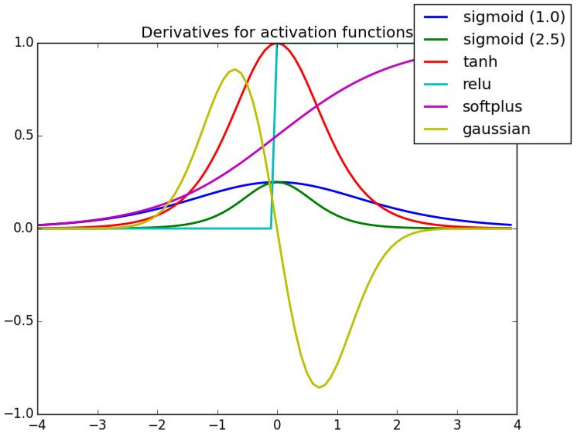
- Weight Multiplication Layers (conv, dense, deconv, groupConv, etc)
- Activation Layers (ReLU, sigmoid, tanh, etc)
- Sampling layers (maxpool, avgpool, unpool, etc)
- Combination Layers (concat, skip connections, etc)
- Input Layers (input normalization/shaping/processing)
- Output Layers (classification-softmax, regression-sigmoid, etc)
- Utility layers (dropout, batch-norm, etc)

We learned about "Activation Functions"



Why ReLU in AlexNet





We learned about “Weight Initialization”

Weight initialization is an important design choice when developing deep learning neural network models.

Historically, weight initialization involved using small random numbers, although over the last decade, more specific heuristics have been developed that use information, such as the type of activation function that is being used and the number of inputs to the node.

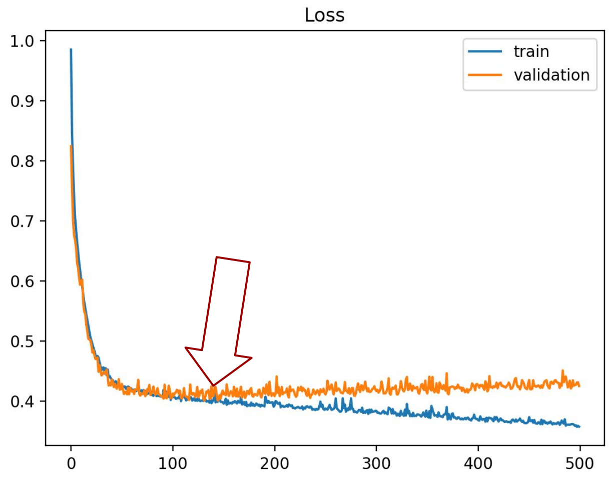
These more tailored heuristics can result in more effective training of neural network models using the stochastic gradient descent optimization algorithm.

We learned about “Training Graphs”

Overfitting

- A plot of learning curves shows overfitting if:
  - The plot of training loss continues to decrease with experience.
  - The plot of validation loss decreases to a point and begins increasing again.

- The *inflection point* in validation loss may be the point at which training could be halted as experience after that point shows the dynamics of overfitting.

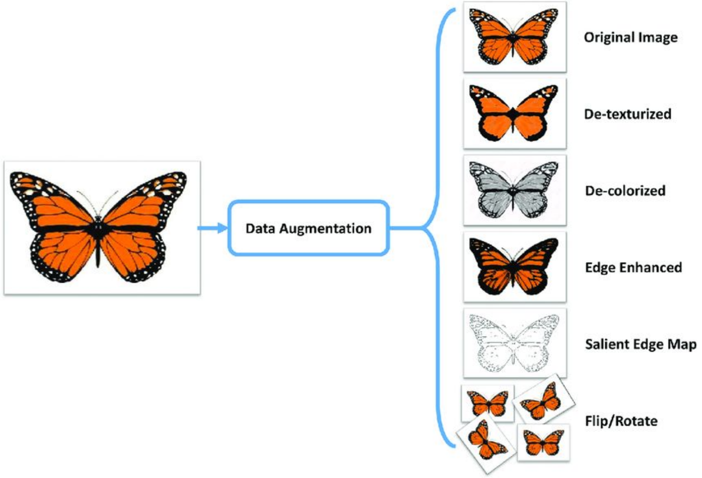


We learned about “Data Augmentation”

The augmentation technique to be applied depends on:

- Input modality (image, sound, text, etc.)
- The problem (classification, segmentation, machine translation, etc.)

Vision:



We learned about "Transfer Learning"

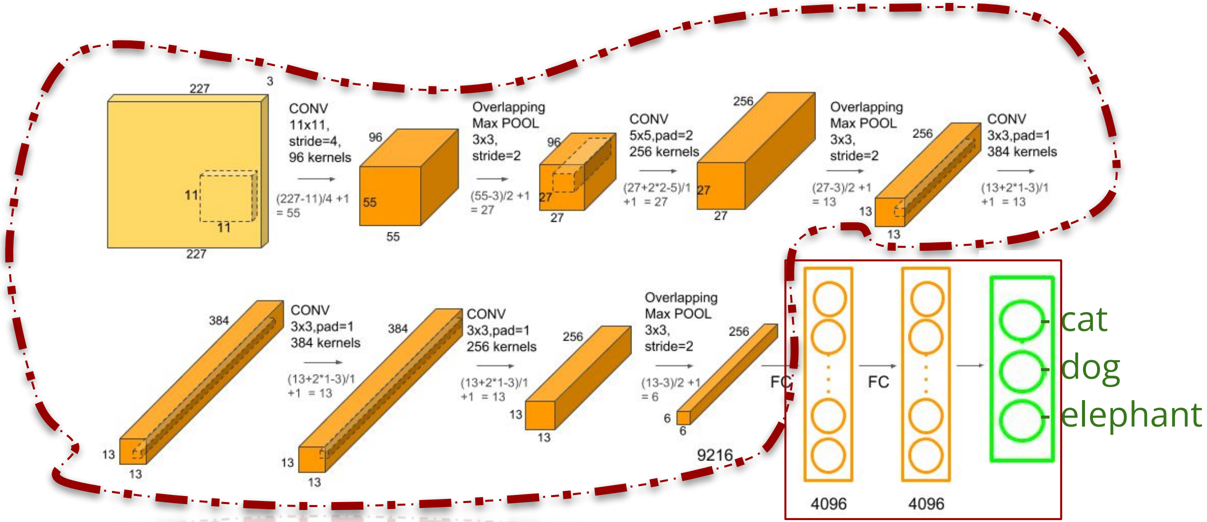
But I don't have enough cat, dog, elephant images !

Remember AlexNet

Maybe we can change the final layer (1000 nodes for 1000 categories) to a 3 node layer to classify

- cat
- dog
- elephant, images

(if that is what we are up to).

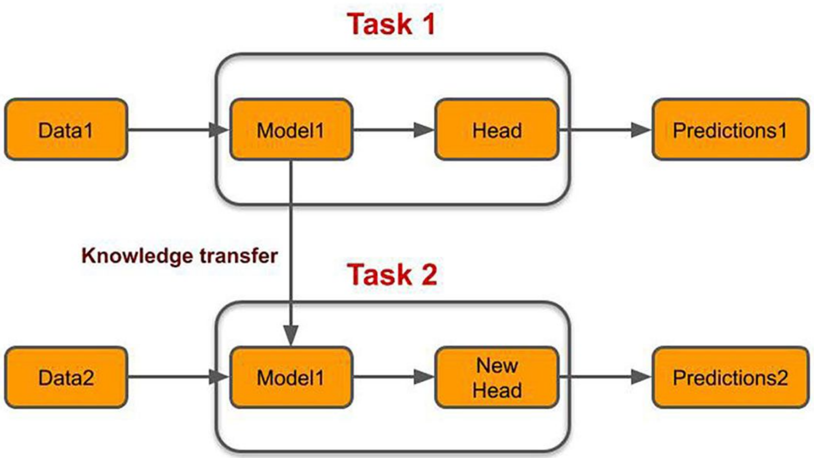


Weights fine-tuned

Weights re-calculated

We learned about “Transfer Learning”

- Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.



We learned about "Sequences"

- So, in humans (unlike FNNs), *the output of one data point is* ***NOT independent*** *of the previous input.*
- A **sequence** can be thought of as a list of elements with a **particular** order.
- Sequence Modeling is the task of modelling what word/letter/element comes next.
- Unlike the FNN and CNN, in sequence modeling, the current output is dependent on the previous input and the length of the input is not fixed.

We learned about “Sequences”

- Sequence modeling, put simply, is the process of modelling/predicting/generating a sequence of values by analyzing a series of previous input values.
- Unlike the FNN and CNN, in sequence modeling,
  1. the current <u>output is dependent on the previous input</u>
  2. and the <u>length of the input is not fixed.</u>

We learned about “Vanilla RNN vs Standard FNN”

- A vanilla RNN is a very similar architecture, but with an additional feedback connection

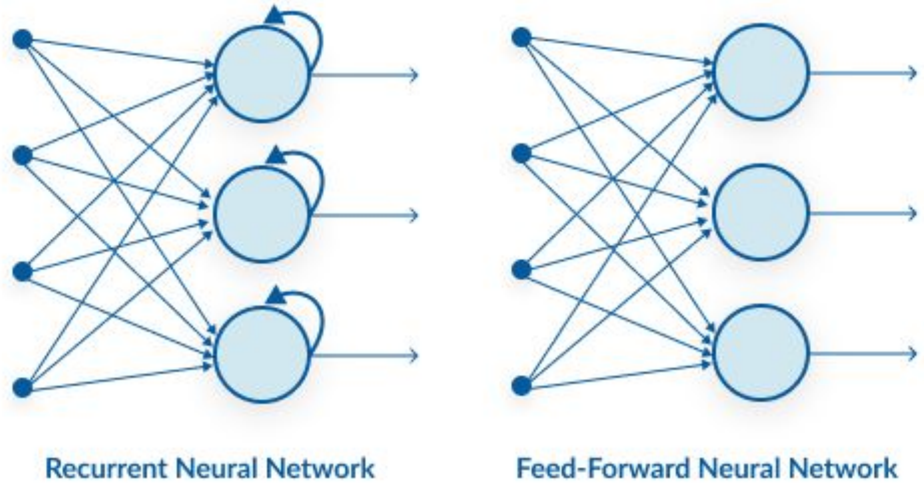
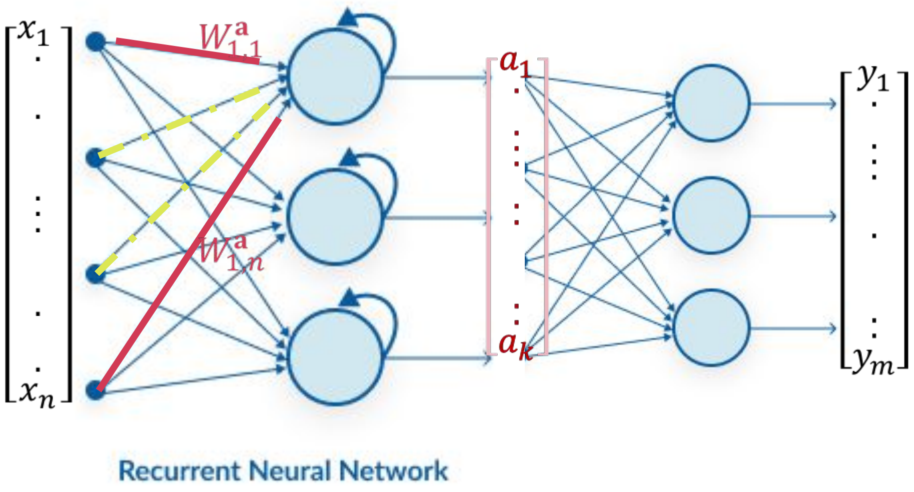


Figure from Aravind Pai, «CNN vs. RNN vs. ANN»

We learned about “Vanilla RNN”

- Now, what is different in RNNs?

This is the general model of a so-called “Vanilla RNN”



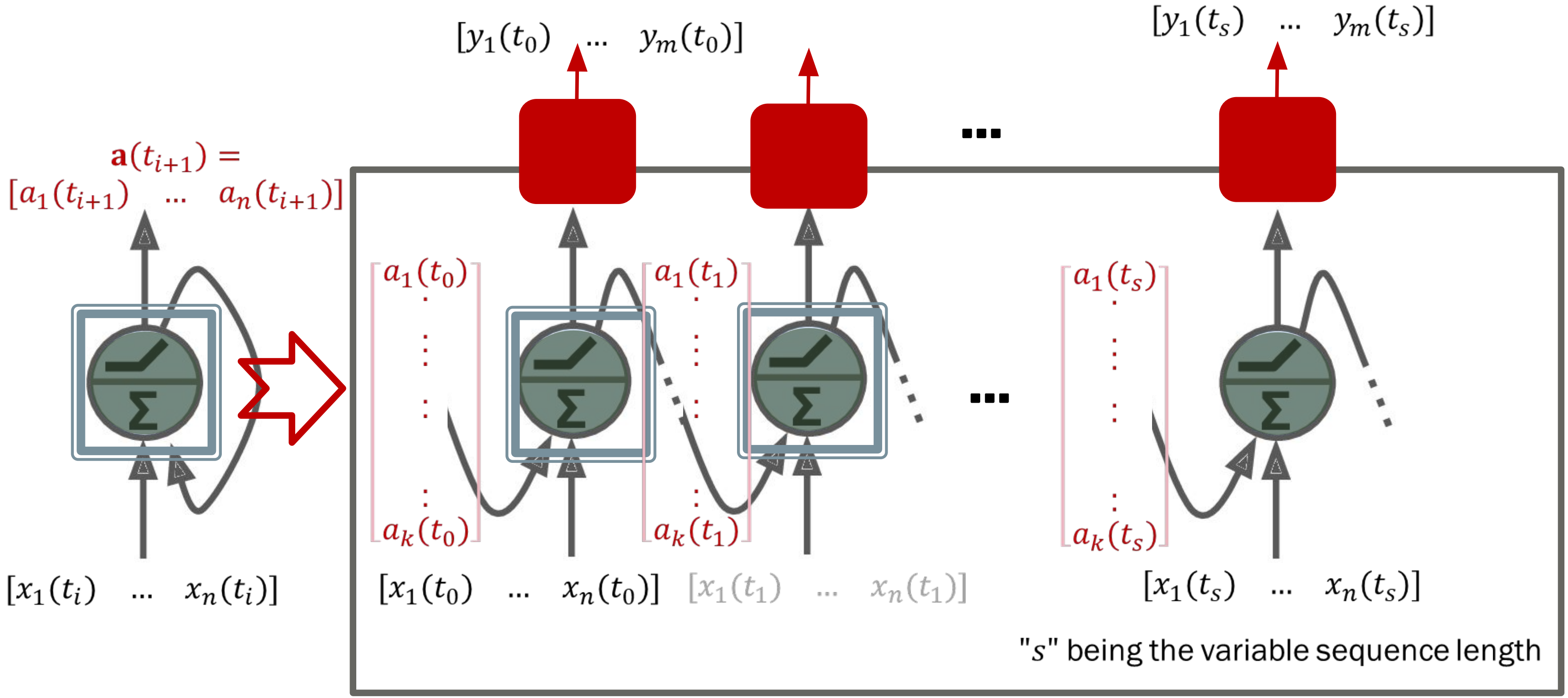
$$\mathbf{a}^{next}_{\mathrm{k}\times 1} = g\left(\mathbf{W}^{\mathbf{a}}_{\mathrm{k}\times(\mathrm{k}+\mathrm{n})} \cdot \begin{bmatrix} \mathbf{x}^{prev}_{\mathrm{n}\times 1} \\ \mathbf{a}^{prev}_{\mathrm{k}\times 1} \end{bmatrix} + \mathbf{b}^{\mathbf{a}}_{\mathrm{k}\times 1}\right)$$

$$\mathbf{W}^{\mathbf{a}}_{\mathrm{k}\times(\mathrm{k}+\mathrm{n})} = [\mathbf{W}^{\mathbf{ax}}_{\mathrm{k}\times \mathrm{n}} \quad \mathbf{W}^{\mathbf{aa}}_{\mathrm{k}\times \mathrm{k}}]$$

$$\mathbf{a}^{next}_{\mathrm{k}\times 1} = g\left(\mathbf{W}^{\mathbf{ax}}_{\mathrm{k}\times \mathrm{n}} \cdot \mathbf{x}^{prev}_{\mathrm{n}\times 1} + \mathbf{W}^{\mathbf{aa}}_{\mathrm{k}\times \mathrm{k}} \cdot \mathbf{a}^{prev}_{\mathrm{k}\times 1} + \mathbf{b}^{\mathbf{a}}_{\mathrm{k}\times 1}\right)$$

$$\mathbf{y}_{\mathrm{m}\times 1} = g\left(\mathbf{W}^{\mathbf{y}}_{\mathrm{m}\times \mathrm{k}} \cdot \left[\mathbf{a}^{next}_{\mathrm{k}\times 1}\right] + \mathbf{b}^{\mathbf{y}}_{\mathrm{m}\times 1}\right)$$

We learned about «unrolling in time»



We learned about "Sequence model I/O Types"

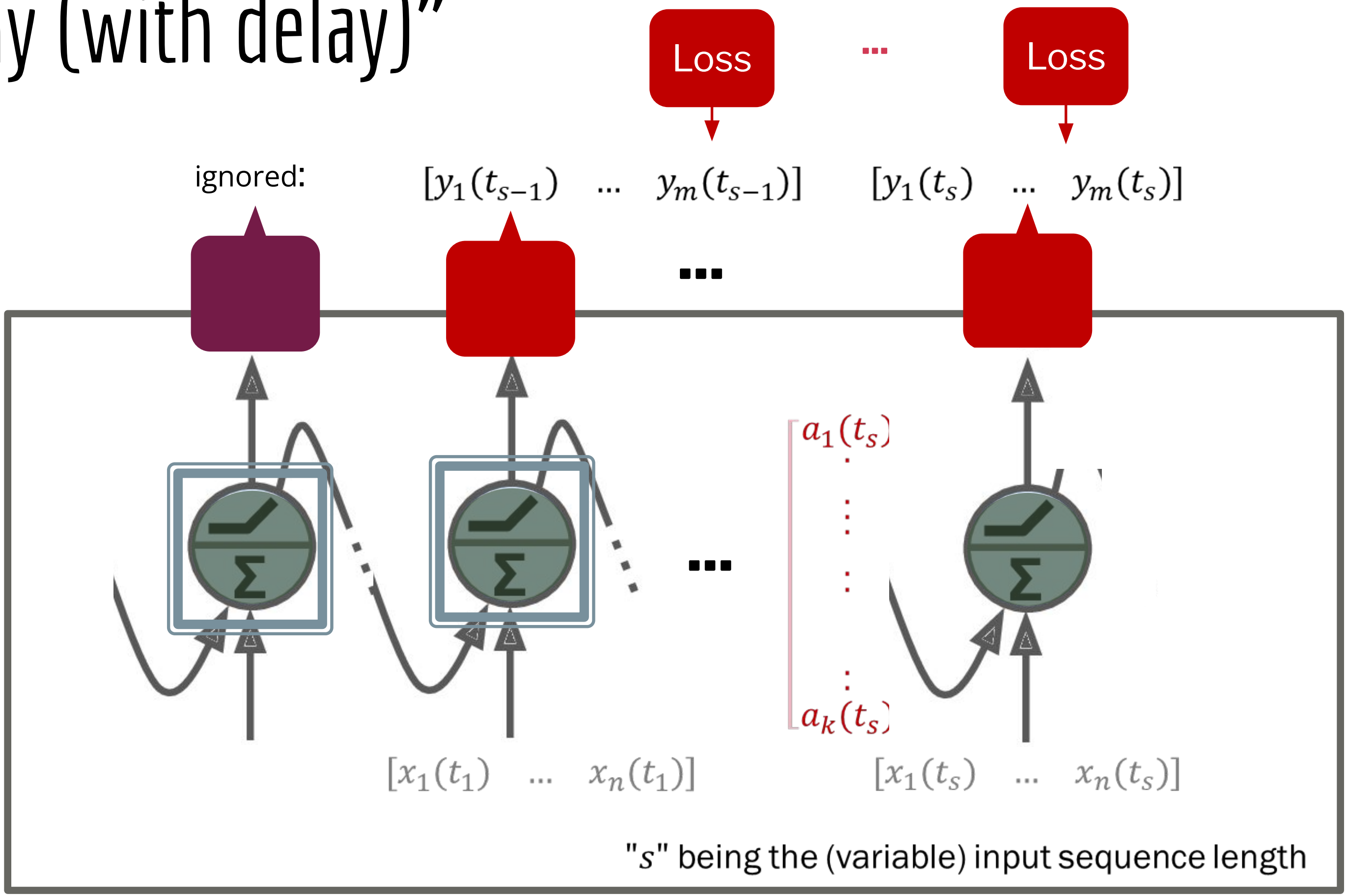
- With simple modifications on this «*many-to-many*» model, we may be able to create:

  - one-to-one
  - one-to-many
  - one-to-one (with-delay)
  - many-to-one
  - many-to-many (with delay)

  models, as well!

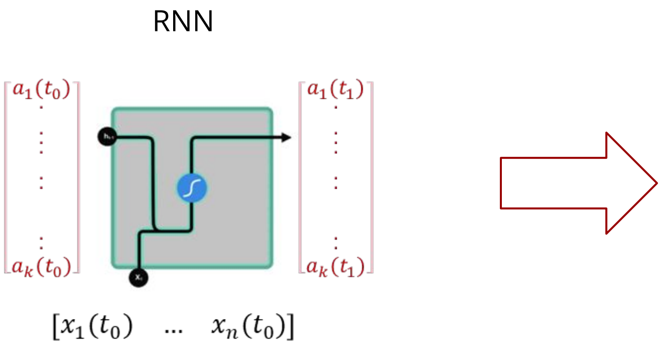
"Many-to-Many (with delay)"

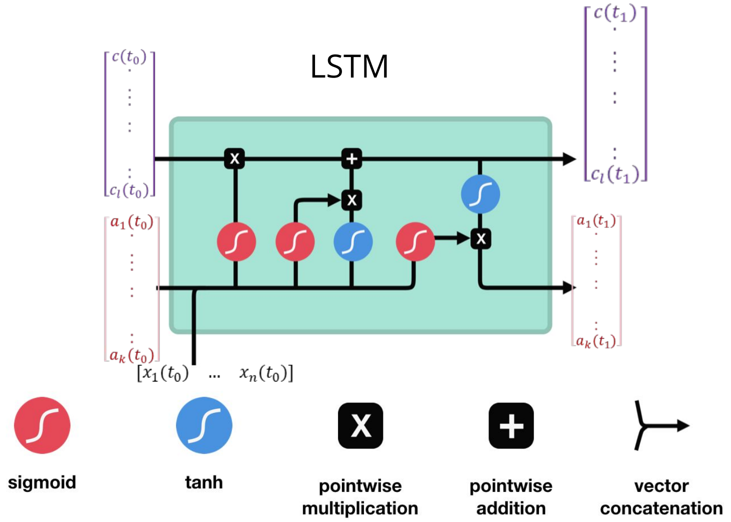
- Not only the last, but some other recent output is also used in training.
- Example: ?



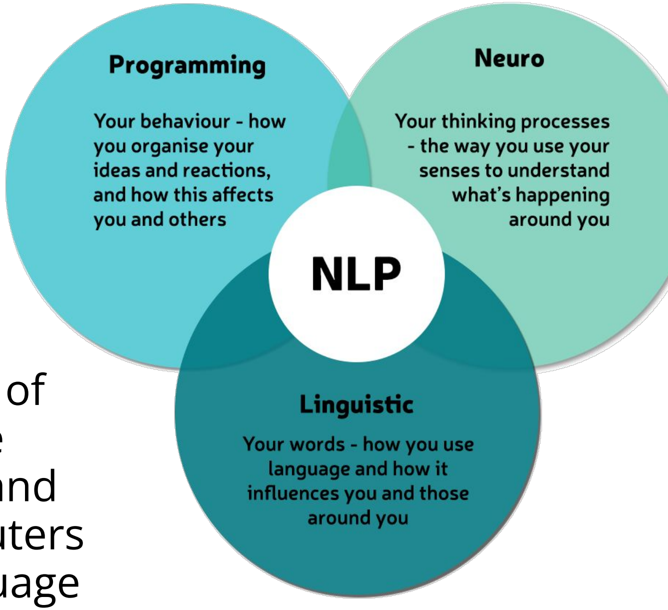
We learned about "LSTM"

- An LSTM has a similar control flow as a recurrent neural network. It processes data passing on information as it propagates forward. The differences are the operations within the LSTM's cells.





We introduced “NLP”



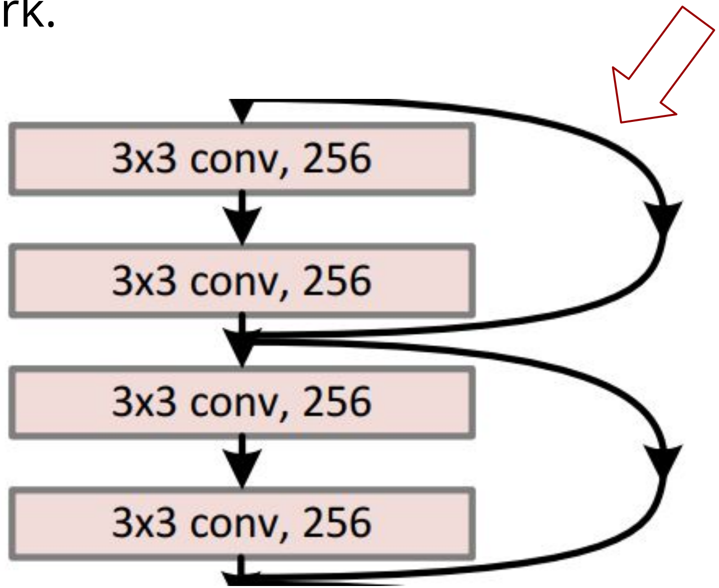
- NLP is Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data.

- The result is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them.
- The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves.

METU

We learned about “Residual Networks”

- They performed a simple set, where they introduced a type of skip connection to a serial (VGG like) network.



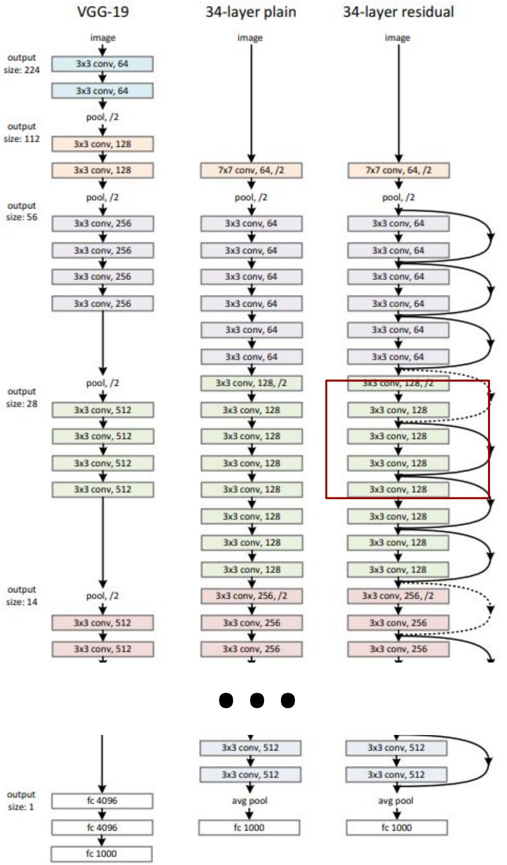
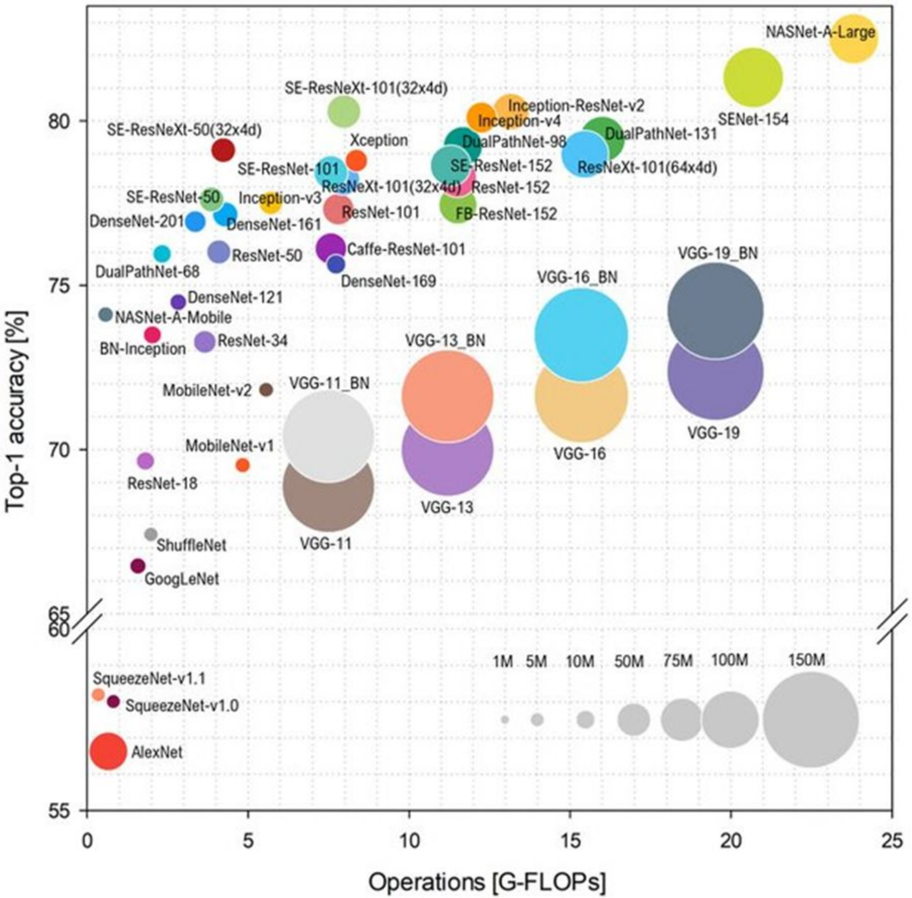


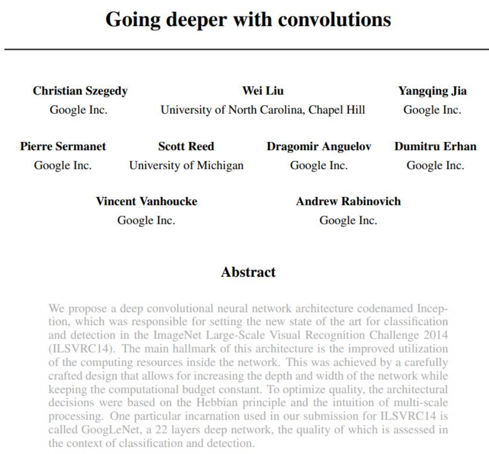
Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Classification Networks

- There are many classification studies that succeeded AlexNet and VGG.
- All trying to optimize computation, memory and accuracy.



We learned about “Inception Modules”

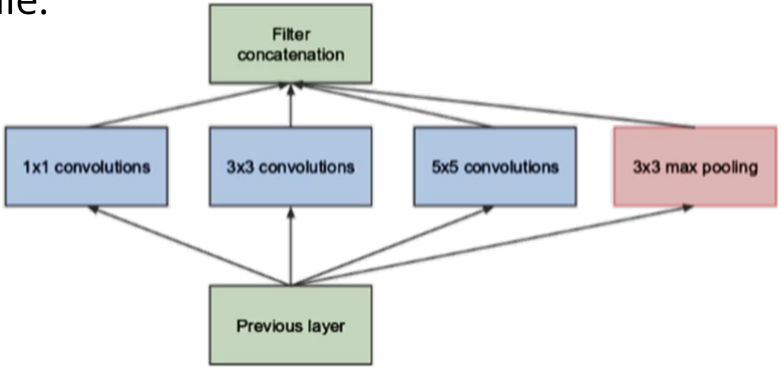

Going deeper with convolutions

Christian Szegedy, Google Inc.
Wei Liu, University of North Carolina, Chapel Hill
Yangqing Jia, Google Inc.
Pierre Sermanet, Google Inc.
Scott Reed, University of Michigan
Dragomir Anguelov, Google Inc.
Dumitru Erhan, Google Inc.
Vincent Vanhoucke, Google Inc.
Andrew Rabinovich, Google Inc.

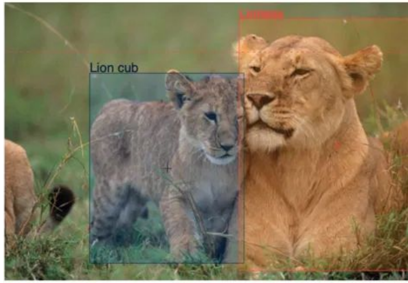
Abstract

We propose a deep convolutional neural network architecture codenamed Inception, which was responsible for setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The main hallmark of this architecture is the improved utilization of the computing resources inside the network. This was achieved by a carefully crafted design that allows for increasing the depth and width of the network while keeping the computational budget constant. To optimize quality, the architectural decisions were based on the Hebbian principle and the intuition of multi-scale processing. One particular incarnation used in our submission for ILSVRC14 is called GoogLeNet, a 22 layers deep network, the quality of which is assessed in the context of classification and detection.

- Why not have filters with multiple sizes operate on the same level?
- The network essentially would get a bit “wider” rather than “deeper”.
- The below image is the “naive” inception module.
  It performs convolution on an input, with 3 different sizes of filters (1x1, 3x3, 5x5).
- Additionally, max pooling is also performed. The outputs are concatenated and sent to the next inception module.

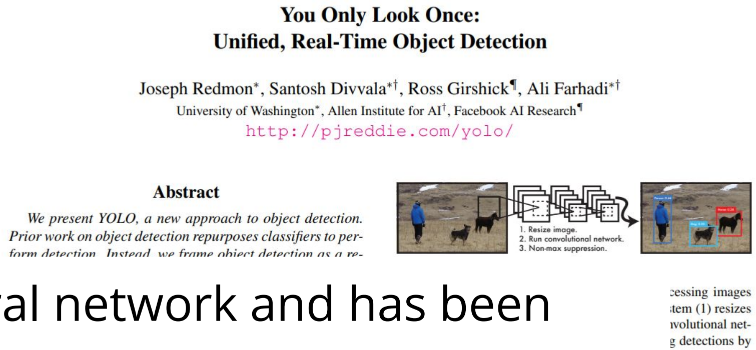


We learned about “Object Detection”



- The object detection network is trained on the annotated data until it can find regions in images that correspond to each kind of object.
- Now let’s look at a few object-detection neural network architectures:
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
  - YOLO
- These architectures are examples that aptly summarize evolutionary development in object detection.

We learned about “YOLO (v1)”


You Only Look Once:
Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a re-

- YOLO is implemented as a convolution neural network and has been evaluated on the PASCAL VOC detection dataset. It consists of a total of 24 convolutional layers followed by 2 fully connected layers.
- The final layer predicts the class probabilities and bounding boxes.

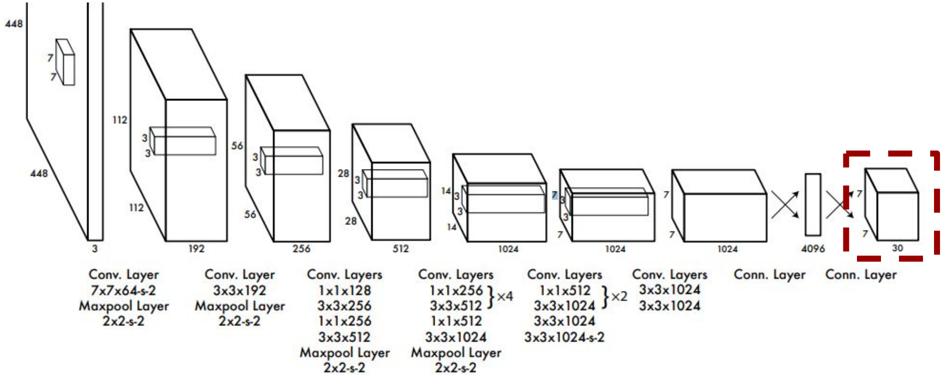
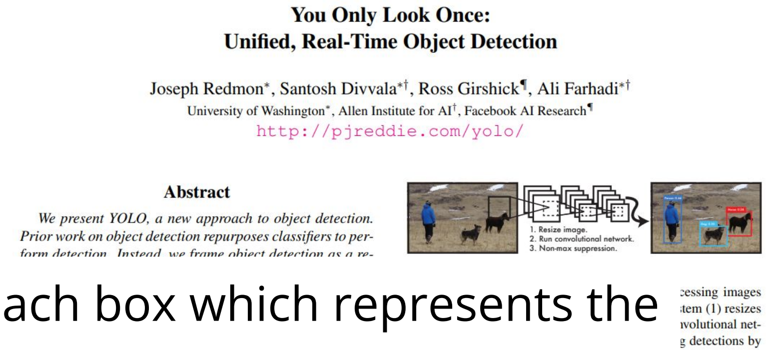


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection.

We learned about “YOLO (v1)”

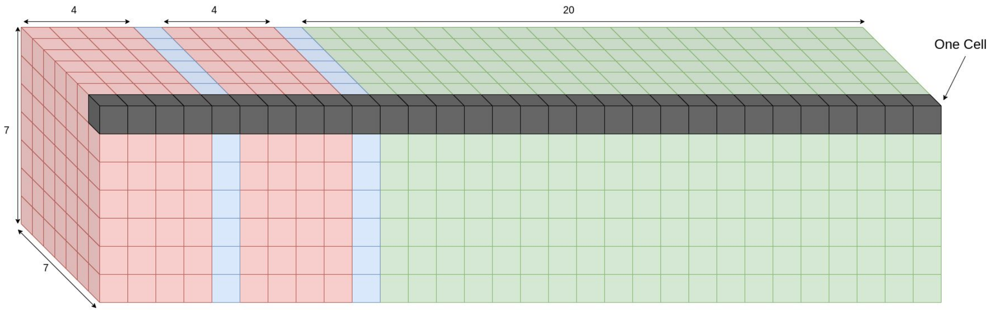

You Only Look Once:
Unified, Real-Time Object Detection

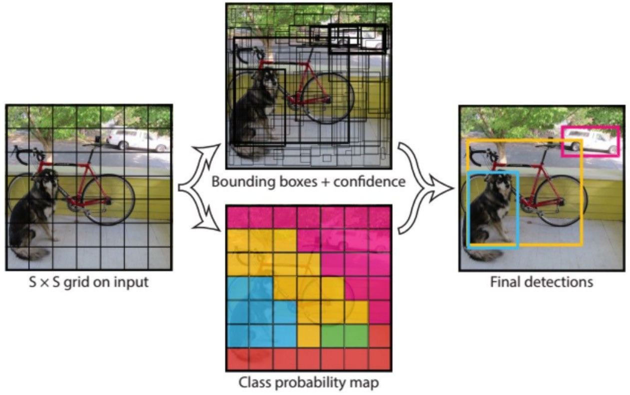
Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

Abstract

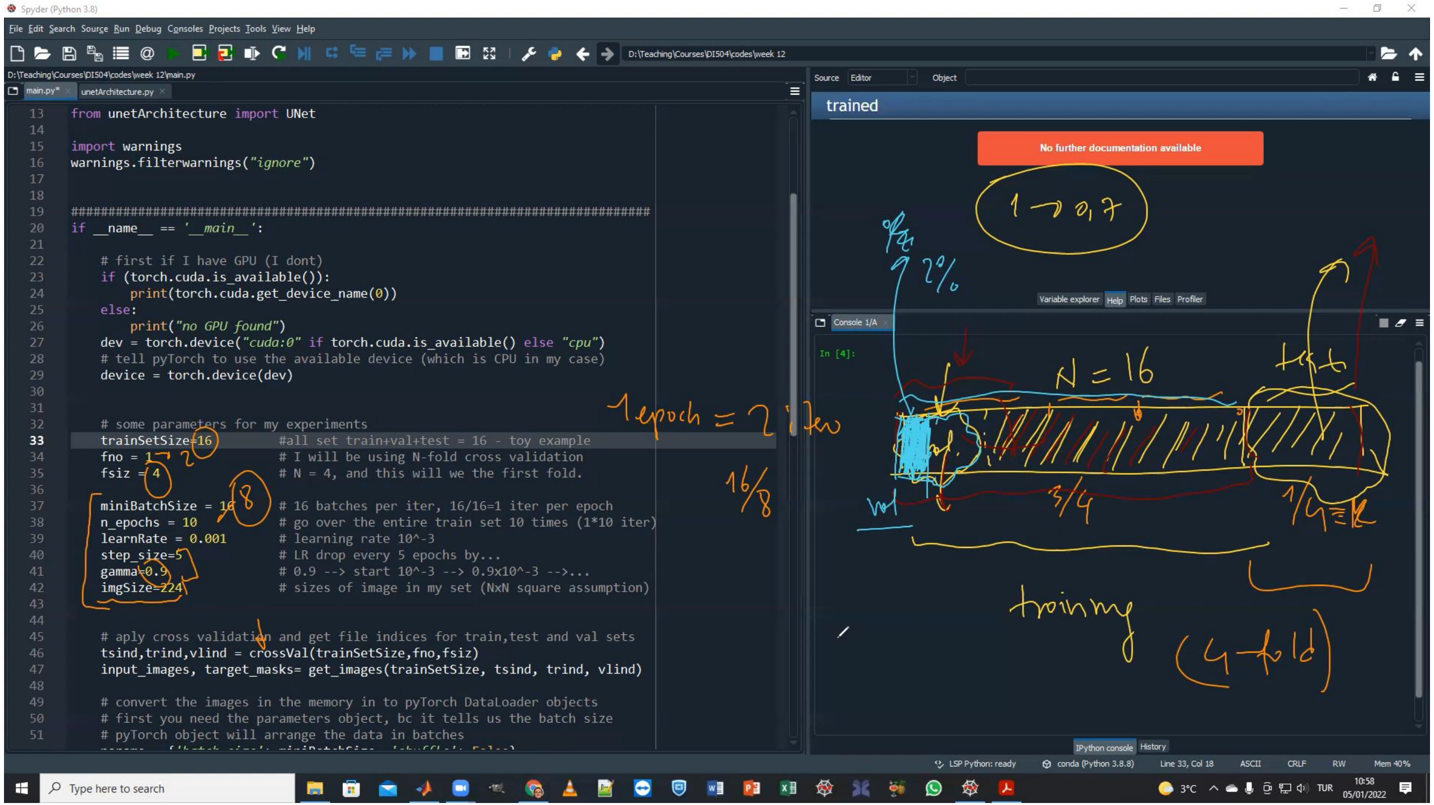
We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to per-

- YOLO also predicts a confidence score for each box which represents the probability that the box contains an object.
- The first five values encode the location and confidence of the first box, the next five encode the location and confidence of the next box, and the final 20 encode the 20 classes (because Pascal VOC has 20 classes).





We live-coded UNET



We live-coded an LSTM network