COMP34212 Lecture
Introduction to Deep Learning

Angelo Cangelosi
Department of Computer Science,
The University of Manchester

Content

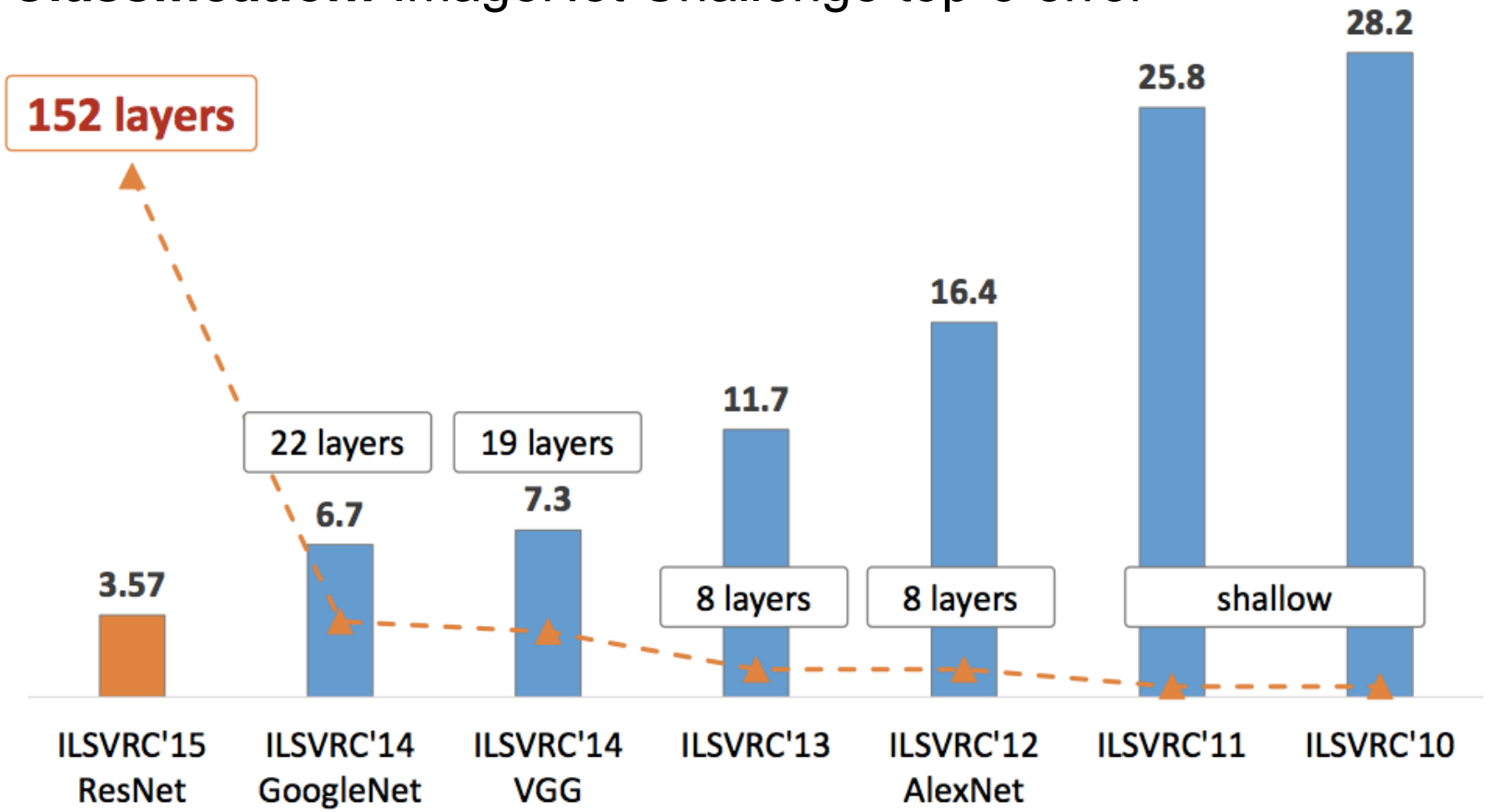
- Why cognitive robotics and computer vision
- Intro to neural networks
  - Definition, Perceptron and MLPs
- Learning algorithms and datasets
  - Backpropagation, training and hyperparameters
- CNNs
- Transformers

Why Cognitive Robotics and (**Cognitive**) Vision?

IMAGENET

ImageNet Challenge

Classification: ImageNet Challenge top-5 error



slazebni.cs.illinois.edu

Article

The International Journal of
Robotics Research
2018, Vol. 37(4–5) 405–420

DOI: 10.1177/0278364918770733
journals.sagepub.com/home/ijr

The limits and potentials of deep learning for robotics

Niko Sünderhauf[1], Oliver Brock[2], Walter Scheirer[3], Raia Hadsell[4], Dieter Fox[5], Jürgen Leitner[1], Ben Upcroft[6], Pieter Abbeel[7], Wolfram Burgard[8], Michael Milford[1] and Peter Corke[1]

Abstract

The application of deep learning in robotics leads to very specific problems and research questions that are typically not addressed by the computer vision and machine learning communities. In this paper we discuss a number of robotics-specific learning, reasoning, and embodiment challenges for deep learning. We explain the need for better evaluation metrics, highlight the importance and unique challenges for deep robotic learning in simulation, and explore the spectrum between purely data-driven and model-driven approaches. We hope this paper provides a motivating overview of important research directions to overcome the current limitations, and helps to fulfill the promising potentials of deep learning in robotics.

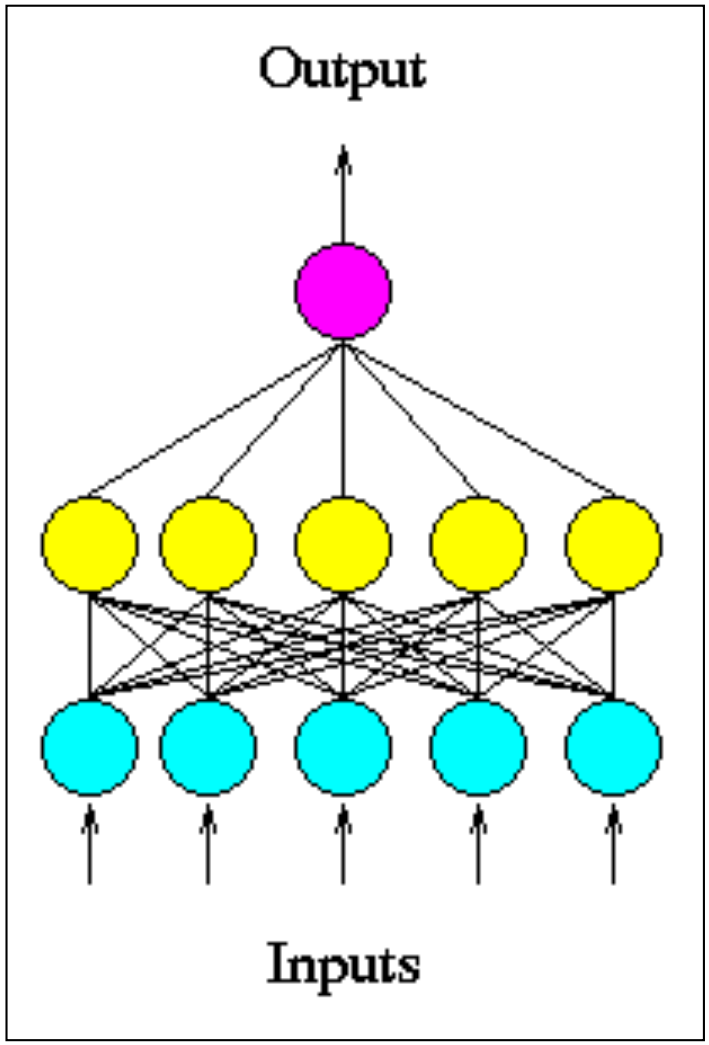
Keywords

Robotics, deep learning, machine learning, robotic vision

What is a Neural Network?

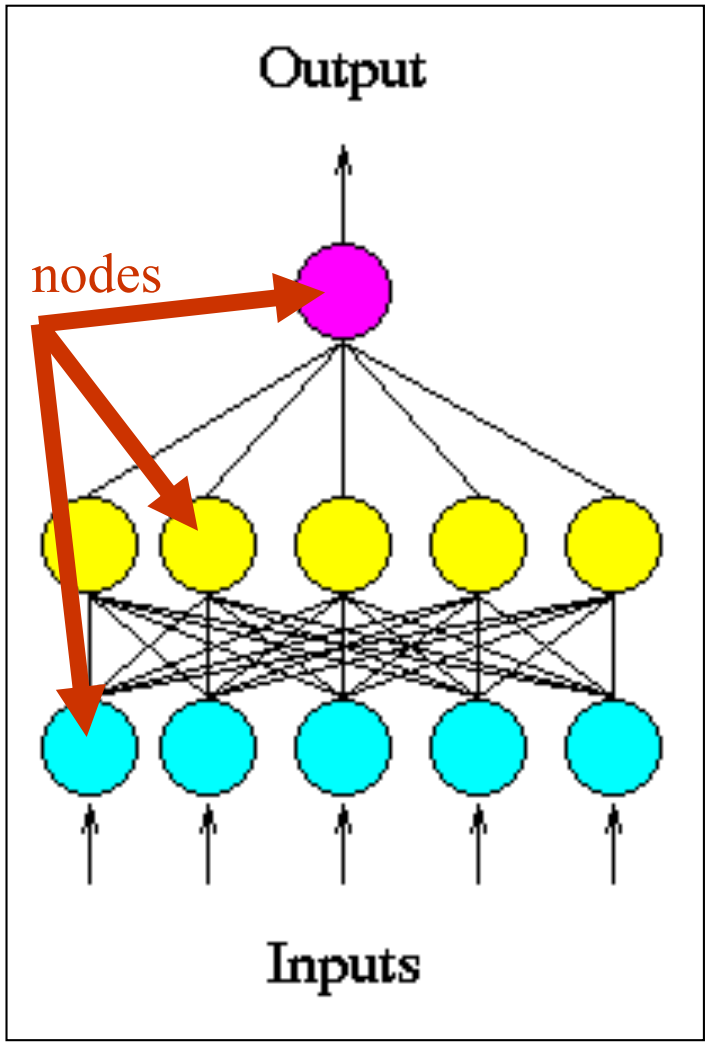
Neural Network

- An artificial neural network consists of a set of interconnected nodes. It is able to receive input stimuli from the environment and to learn new behaviours/responses



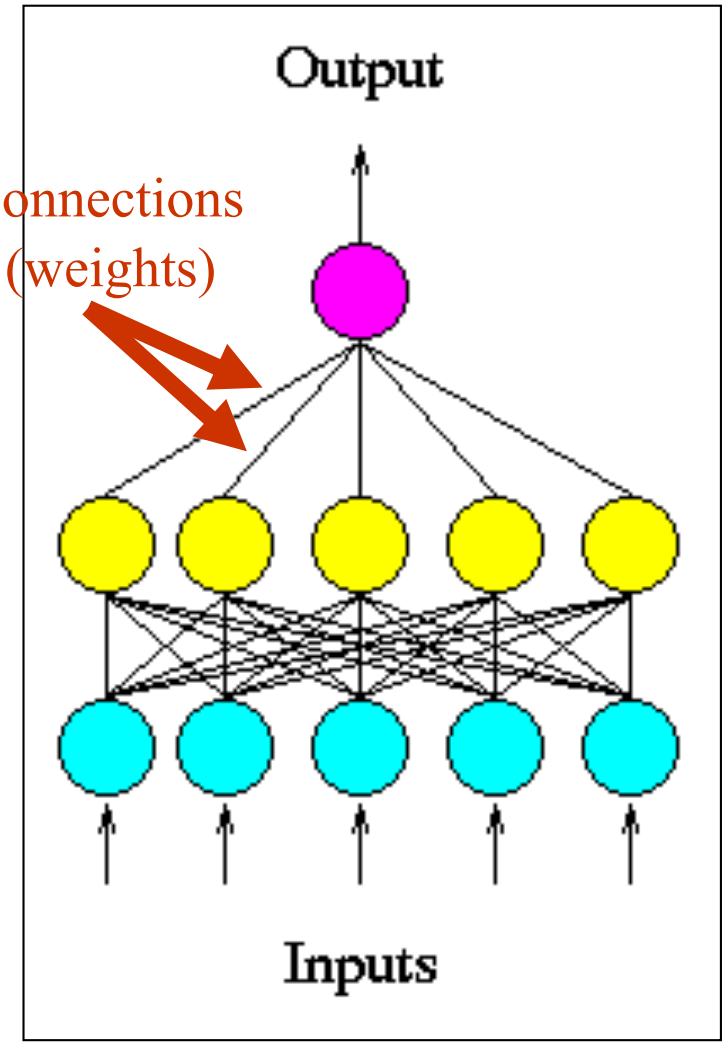
Neural Network

- An artificial neural network consists of a set of interconnected **nodes**. It is able to receive input stimuli from the environment and to learn new behaviours/responses



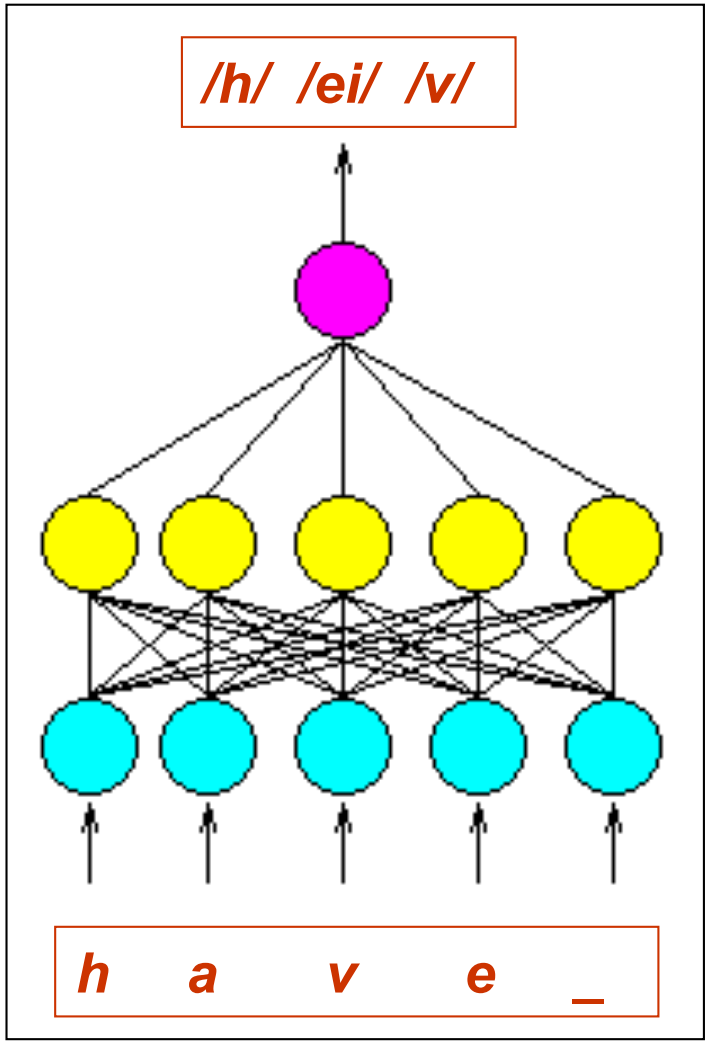
Neural Network

- An artificial neural network consists of a set of **interconnected** nodes. It is able to receive input stimuli from the environment and to learn new behaviours/responses



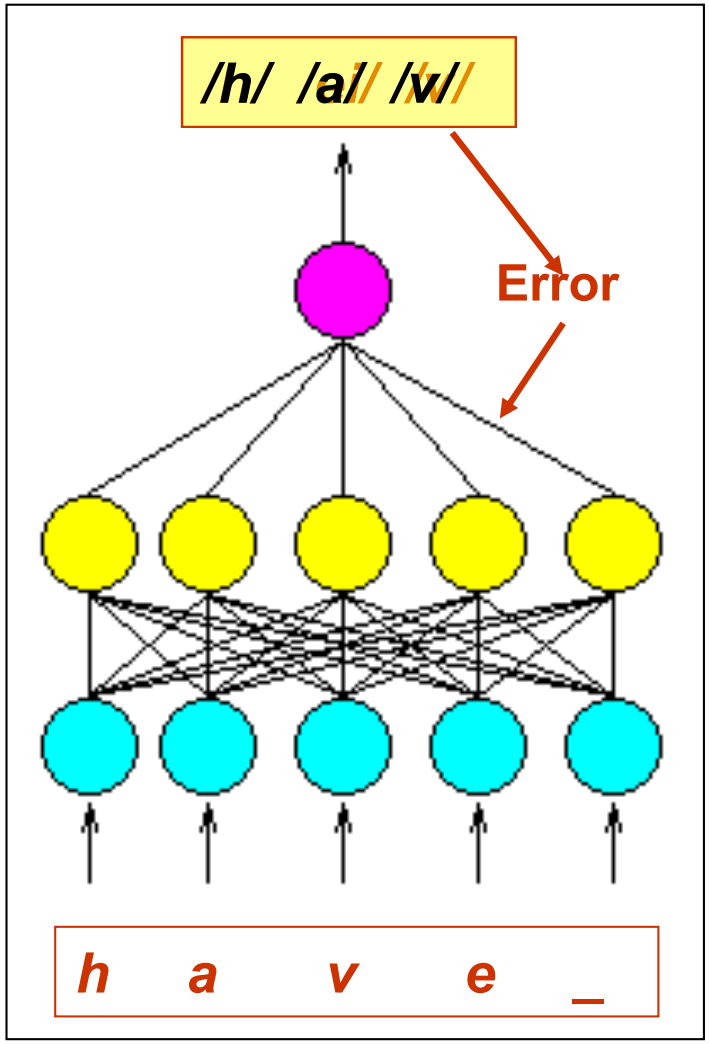
Neural Network

- An artificial neural network consists of a set of interconnected nodes. It is able to receive **input stimuli** from the environment and to learn new behaviours/responses



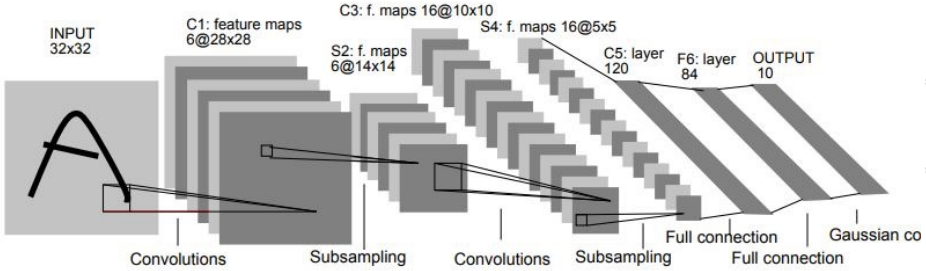
Neural Network

- An artificial neural network consists of a set of interconnected nodes. It is able to receive input stimuli from the environment and **to learn** new behaviours/responses

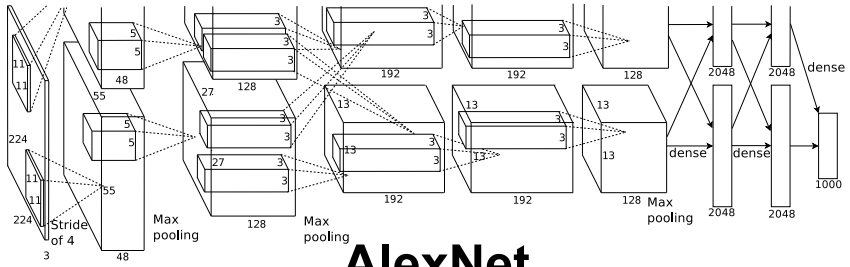


Shallow vs. Deep NN

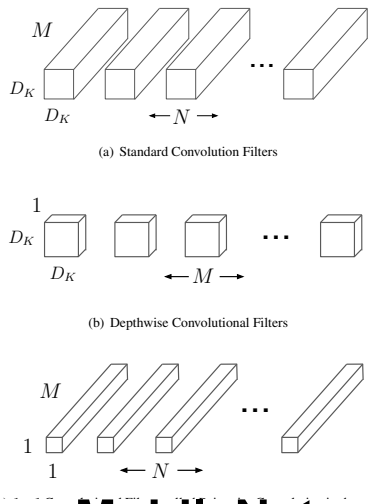
- **Shallow** (Classical) Neural Networks
  - Simple Perceptron (2 Layers)
  - MLP Multi-Payer Perceptron (3 layers)
- **Deep** Neural Networks
  - CNN Convolutional Neural Networks
  - RNN Recurrent Neural Networks (e.g.LSTM)
  - Transformers
  - Other types: GAN, …
  - Celebrities: AlexNet, VGG16, ResNet50, BERT, ViT, GPT…



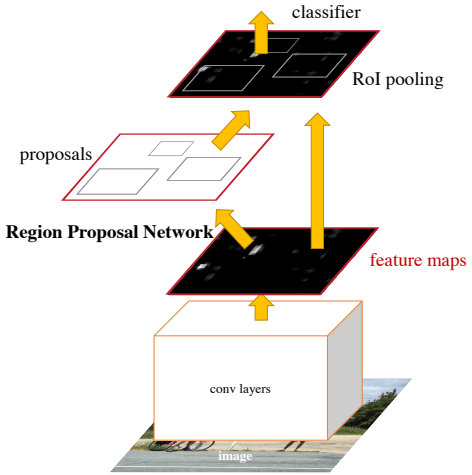
LeNet-5



AlexNet

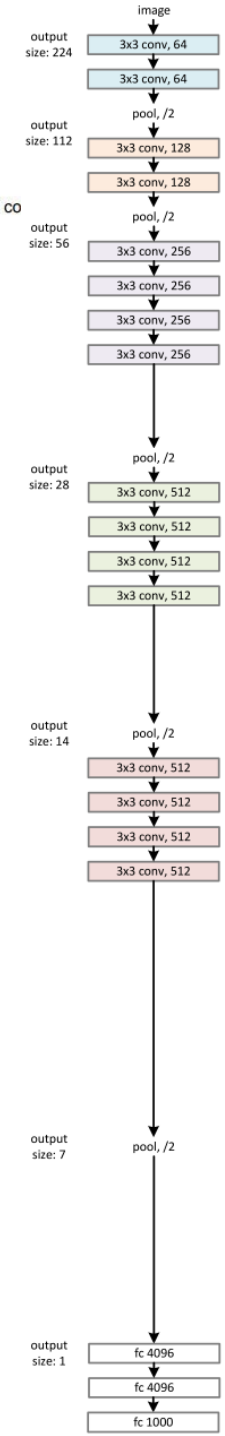


MobileNet

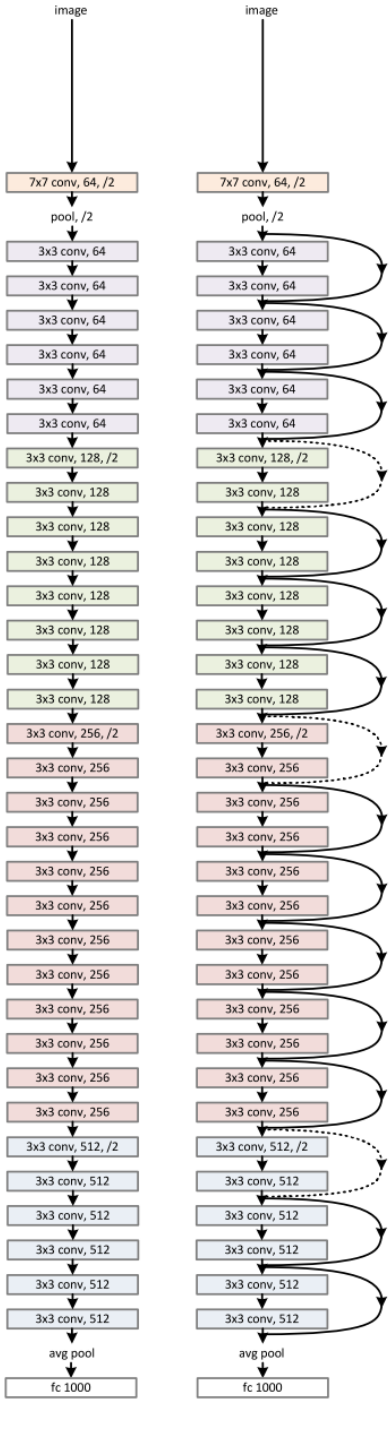


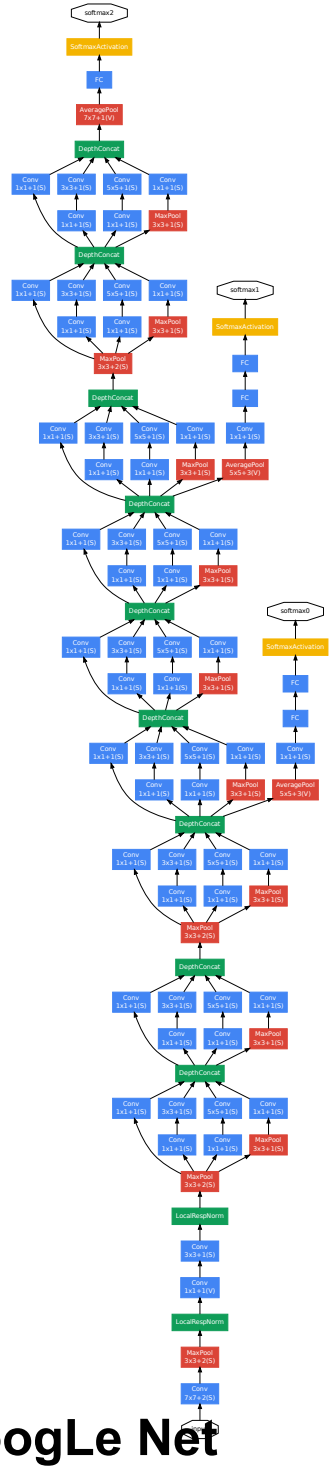
Faster RCNN

VGGNet



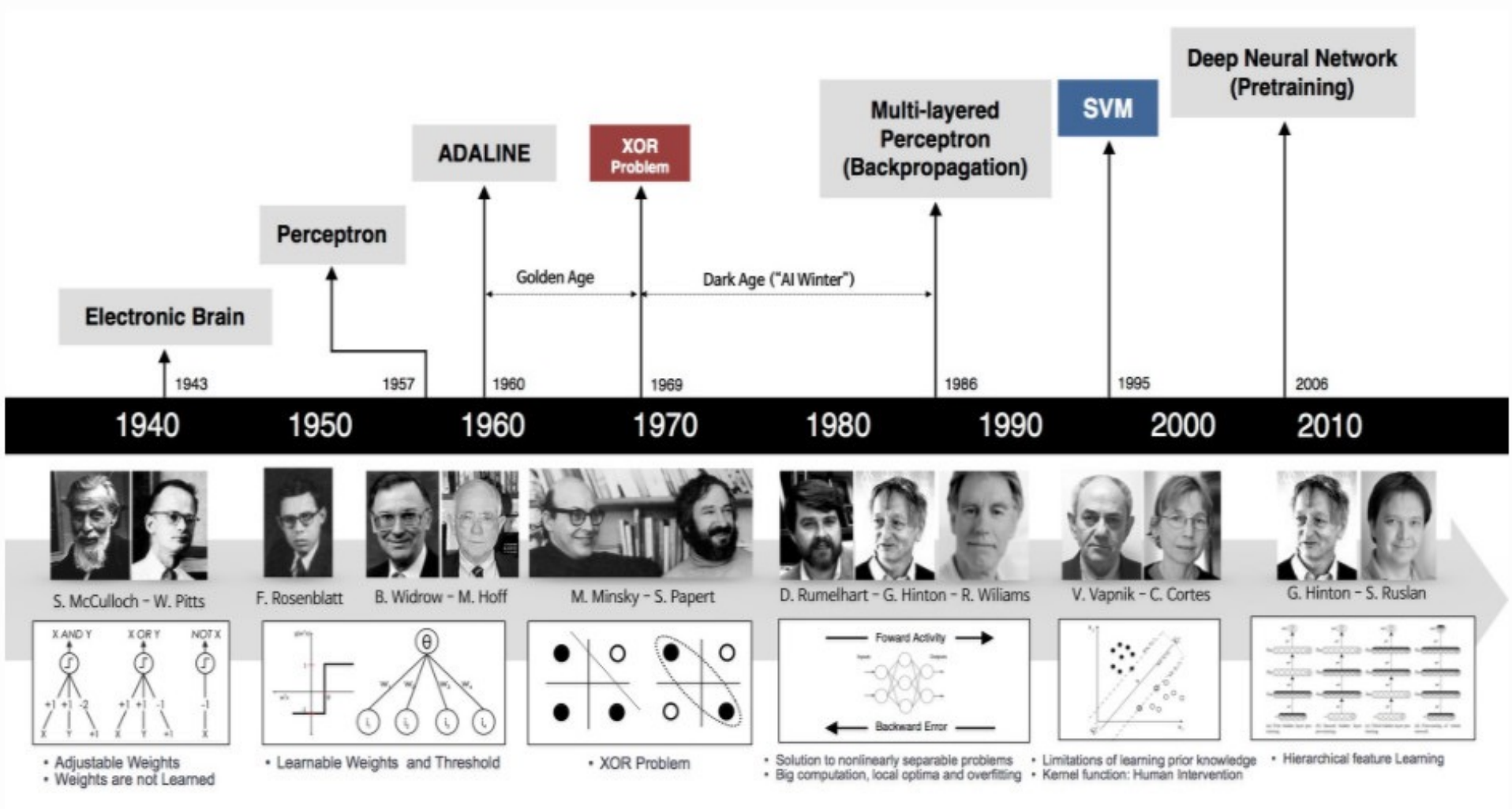
ResNets





GoogLe Net

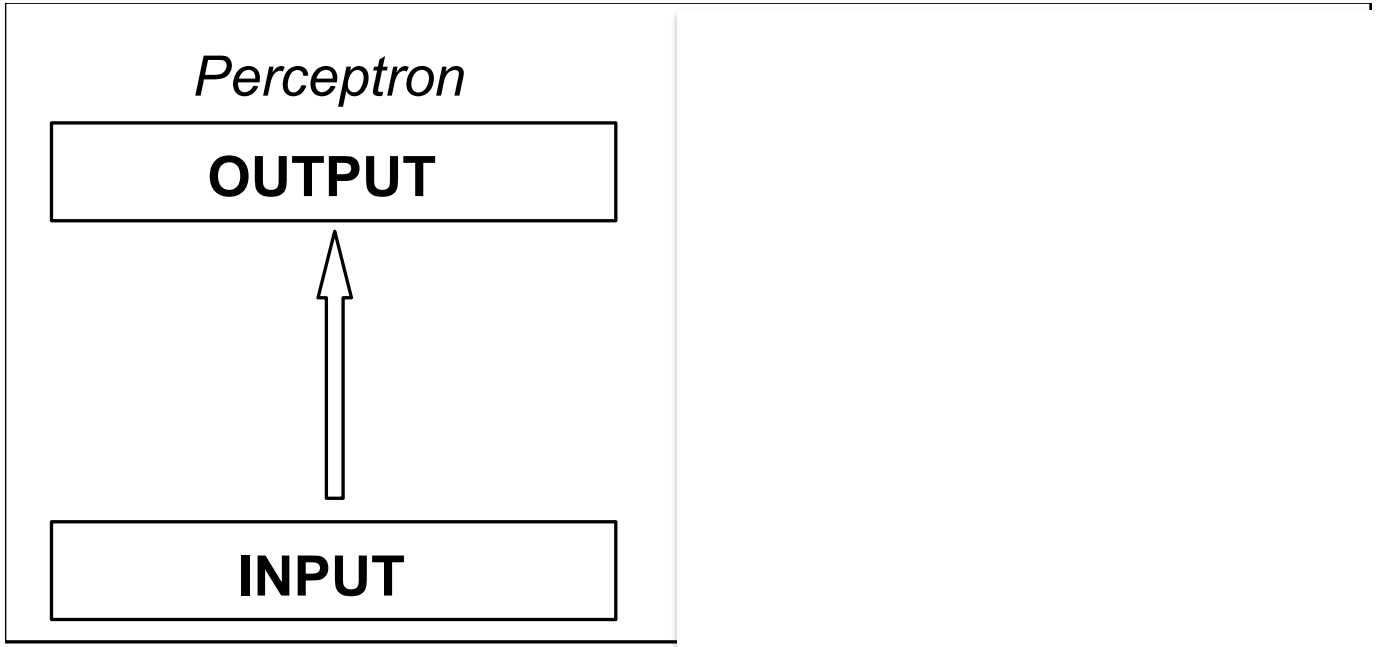
NNet History



(Shallow)
Neural Networks
Topologies

NN Topologies

- Feed-forward models
  - Perceptron



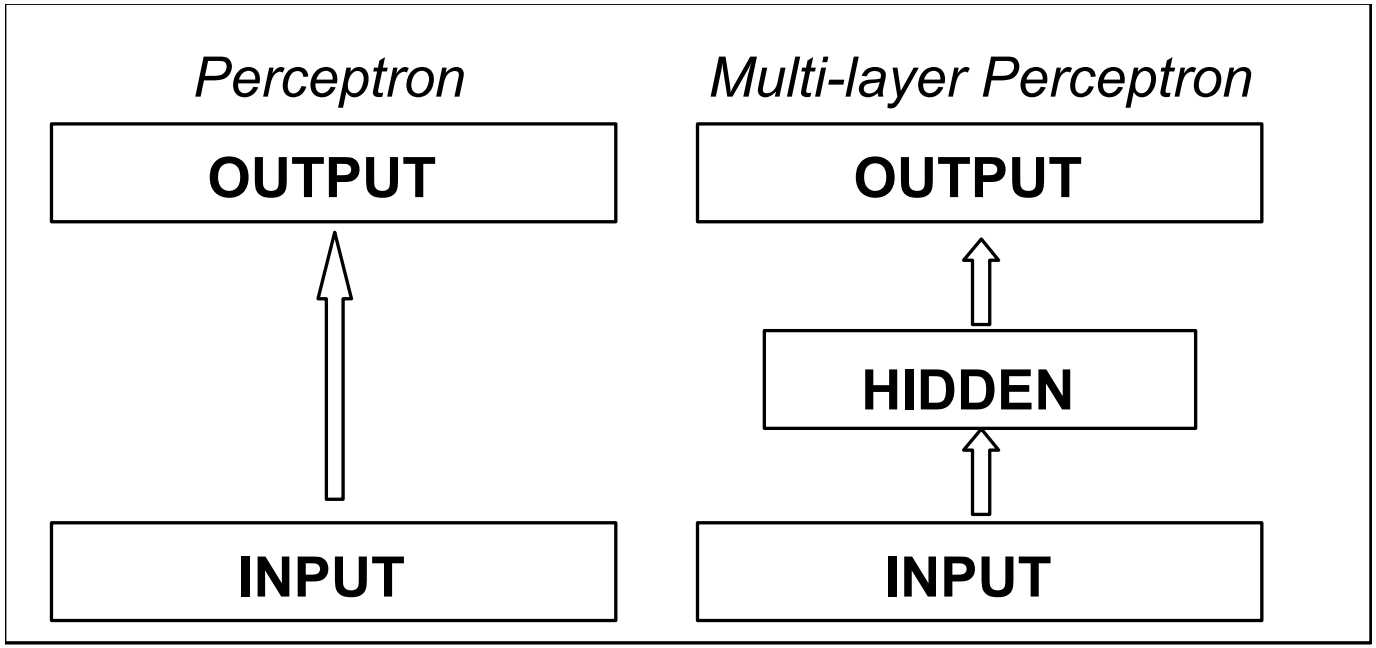
Keras Code Example

- Feed-forward models
  - Perceptron

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential()

# adding 1 layer (the output layer)
model.add(Dense(1, input_dim=2, activation='sigmoid'))

# compilation
model.compile(loss='mean_squared_error', optimizer='sgd', metrics=['accuracy'])

# Trainng (fit) of the model
model.fit(input_data,output_data, epochs=5000, batch_size=4)

# Test of the prediction after training
model.predict(input_data, verbose=1)
```

NN Topologies

- Feed-forward models
  - Perceptron
  - Multi-Layer Perceptron MLP (for backpropagation algorithm)



Keras Code Example

- Feed-forward models
  - Multi-Layer Perceptron MLP

MLP for XOR

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential()

# adding INPUP -> HIDDEN
model.add(Dense(16, input_dim=2, activation='relu'))

# adding HIDDEN -> OUTPUT
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='mean_squared_error', optimizer='sgd', metrics=['accuracy'])
```

Keras Code Example

- Feed-forward models
  - Multi-Layer Perceptron MLP

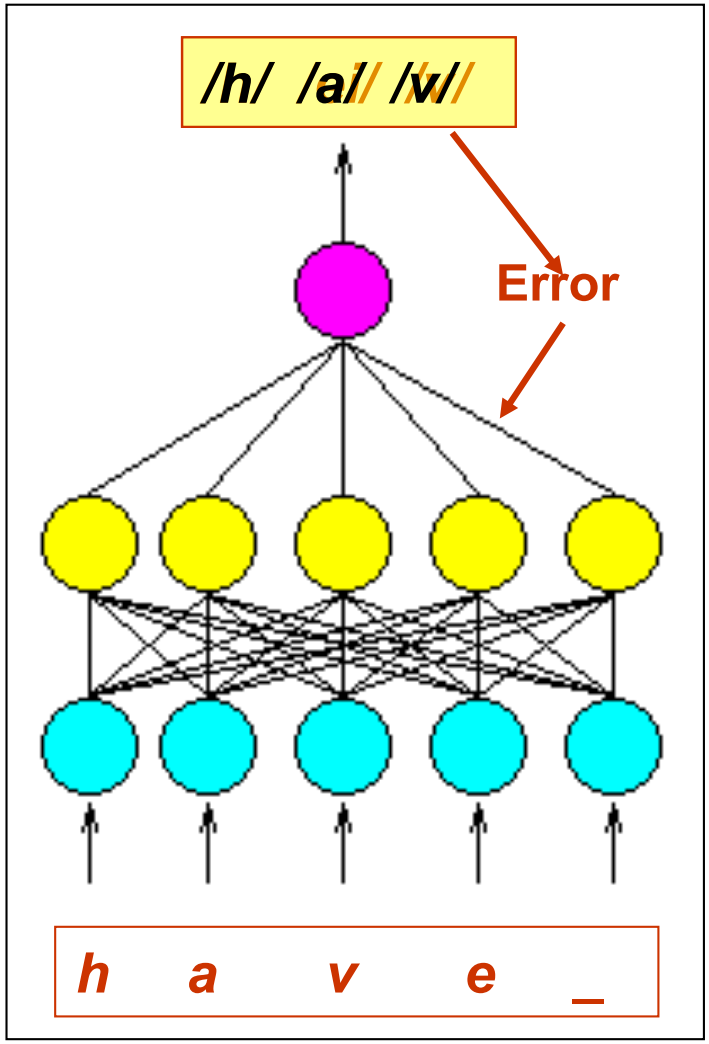
MLP for MNIST

```
N_HIDDEN = 128

model = Sequential()

# Hidden layer 1 with 128 hidden units and ReLu activation function
model.add(Dense(N_HIDDEN, input_shape=(RESHAPED,)))
model.add(Activation('relu'))
# Hidden layer 2 with 128 hidden units and ReLu activation function
model.add(Dense(N_HIDDEN))
model.add(Activation('relu'))

# output layer with 10 units and softmax activation
model.add(Dense(N_CLASSES))
model.add(Activation('softmax'))
```

Neural Networks: Learning Algorithms

Neural Network

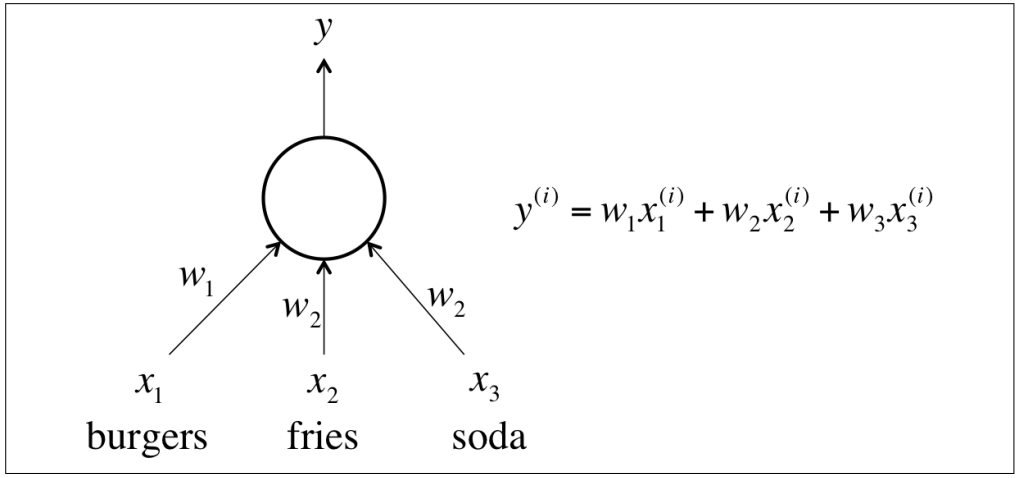
- An artificial neural network consists of a set of interconnected nodes. It is able to receive input stimuli from the environment and **to learn** new behaviours/responses



Training Parameters = Weights

- Training parameters = Find weights
  - To minimize the error E of output $\mathbf{y}$ with respect to target t by iteratively modifying the weights
  - Not solving a set of equations if problem (neuron) is nonlinear!

$$E = \frac{1}{2}\Sigma_i \left(t^{(i)} - y^{(i)}\right)^2$$



Gradient Descend with Nonlinear Neurons

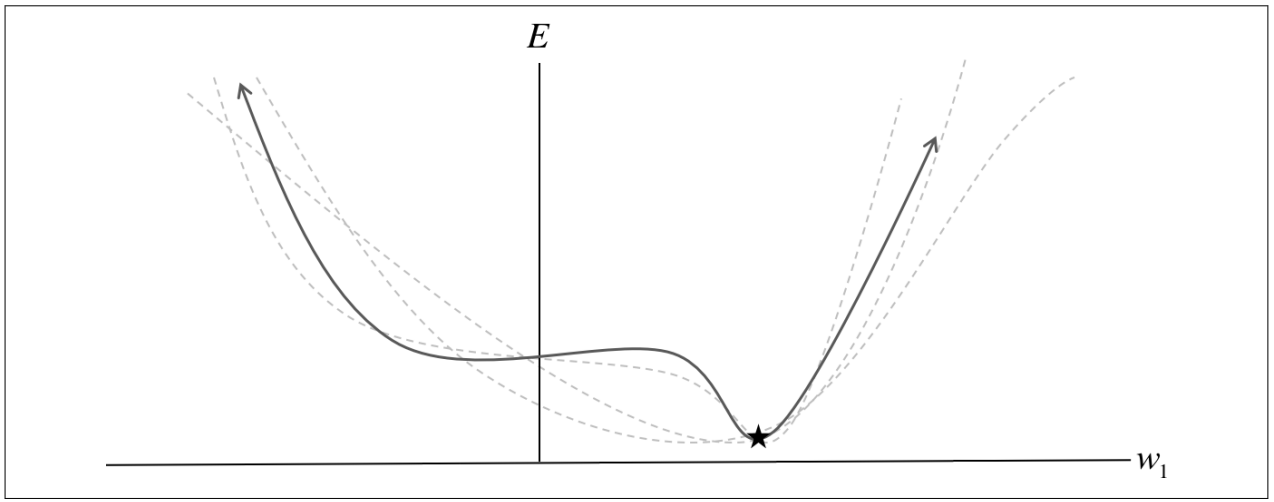
- Error Backpropagation algorithm
- 1986 PDP book (Rumelhart, Hinton & Williams, 1986)

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533

Nature article PDP book chapter

Stochastic Gradient Descent

- Stochastic Gradient Descent
  - Calculate gradient at each iteration/stimulus
  - Dynamic surface

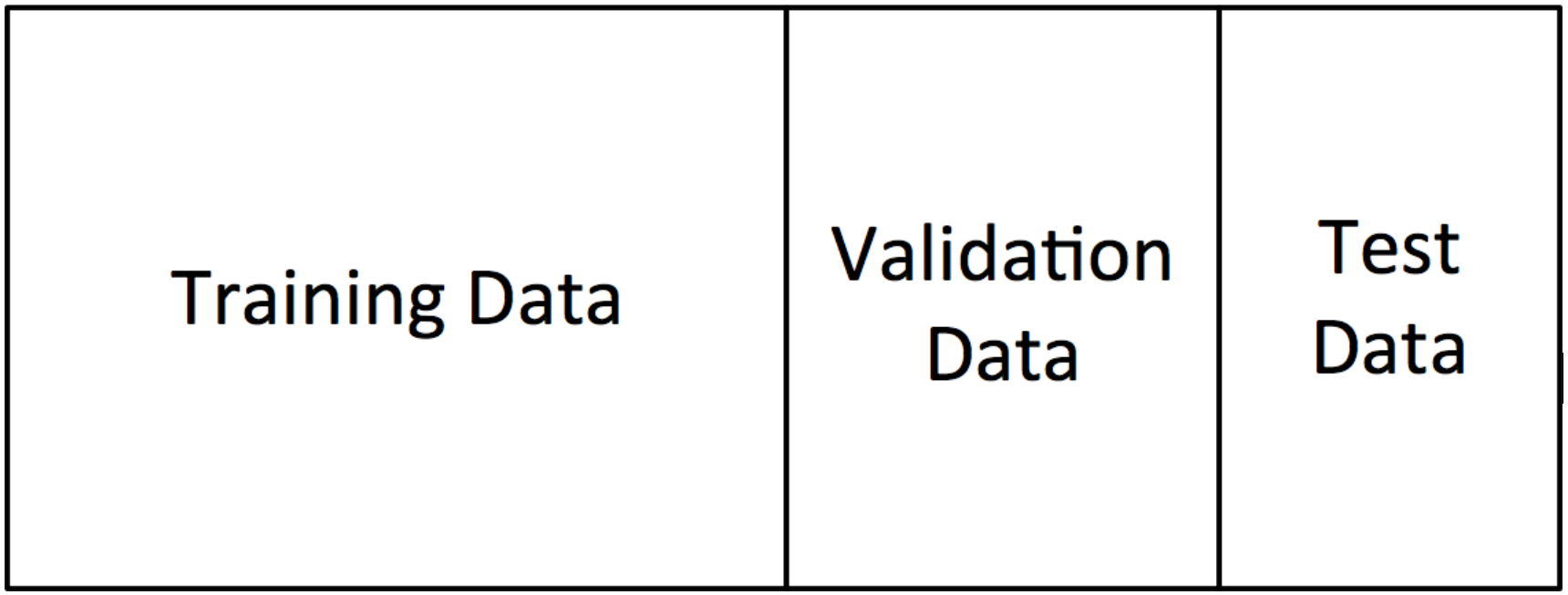


  - But single stimulus error not good approximator
- Mini-Batch Gradient Descent
  - Use Mini-Batch (subset of the total dataset)

Neural Networks: Training and Test

Training Datasets 2

- Full Dataset:

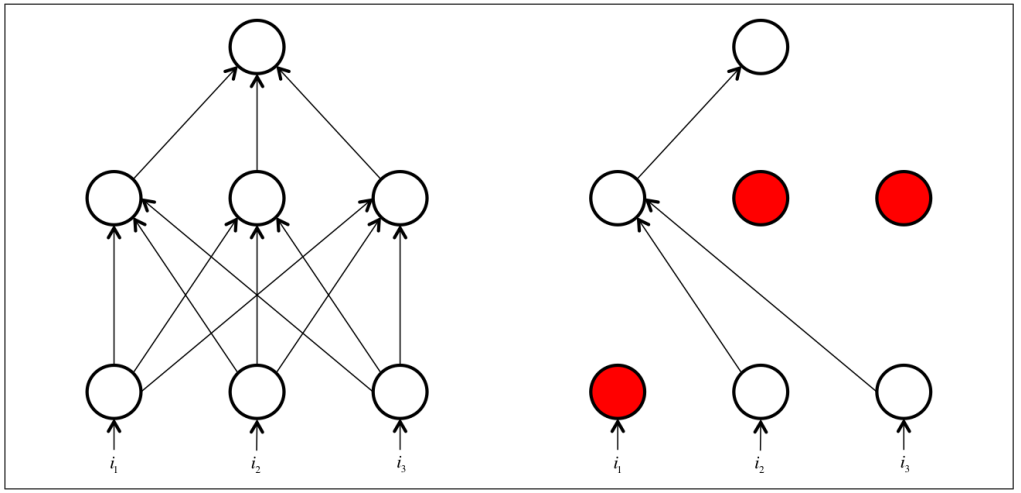


NN Training

- Training Dataset
  - stimuli used, repeatedly, during the training
  - Epoch = use each stimulu **once**
- Validation dataset
  - To monitor test performance during training
  - Set of stimuli set aside (not for training)
- Test Dataset (Generalisation test dataset)
  - Novel stimuli to assess model performance
  - **Overfitting** when model does not generalise well, i.e. test accuracy error is high

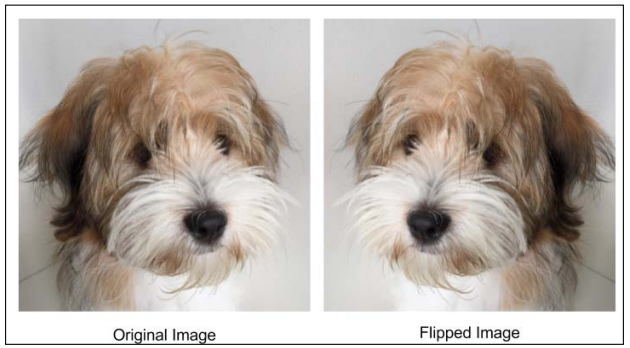
Preventing Overfitting: Dropout

- Dropout
  - Each neuron is inactive with some random probability during each training minibatch



  - Forces the network to be accurate even in the absence of certain information/neurons
  - One of the most used methods in Deep NN

Data Augmentation



- Extend training set generating
  - To reduce overfitting / improve generalisation
  - Generate new image with noise / light changes, shifting position, flipping
  - Data distorsion (LeNet-5)
  - New image translations and horizontal reflections, extracting random 224 × 224 patches (AlexNet)
  - Altering intensities of RGB channels (AlexNet)

Training Terminology

- Error functions (aka Error/Objective funct.)
  - Computes the (average) value of the **error** function
  - **Accuracy:** for classification problems
    - binary accuracy, categorical accuracy, sparse categorical accuracy, top k categorical accuracy
  - **Error loss**: difference between the predicted and the observed values
    - mean square error (MSE), root square error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE), mean squared logarithmic error (MSLE).
  - **Categorical loss**: cross-entropy for classification tasks:
    - binary cross-entropy (2 categories), categorical cross-entropy (multi-class classification), sparse categorical cross-entropy

Hyperparameters

- Network size
  - Number of hidden neurons, number of hidden layers
  - Convolution and pooling parameters (stride, padding…)
- Training parameters
  - Learning rate
  - Momentum
- Optimisation methods SGD, MiniBatch
  - AdaGrad, RMSProp, Adam
- Overfitting
  - Regularisation strength
  - Dropout probability

Neural Networks: Benchmark Datasets

Benchmarking Datasets: MNIST

- 28x28 greyscale
- 60000 training, 10000 test

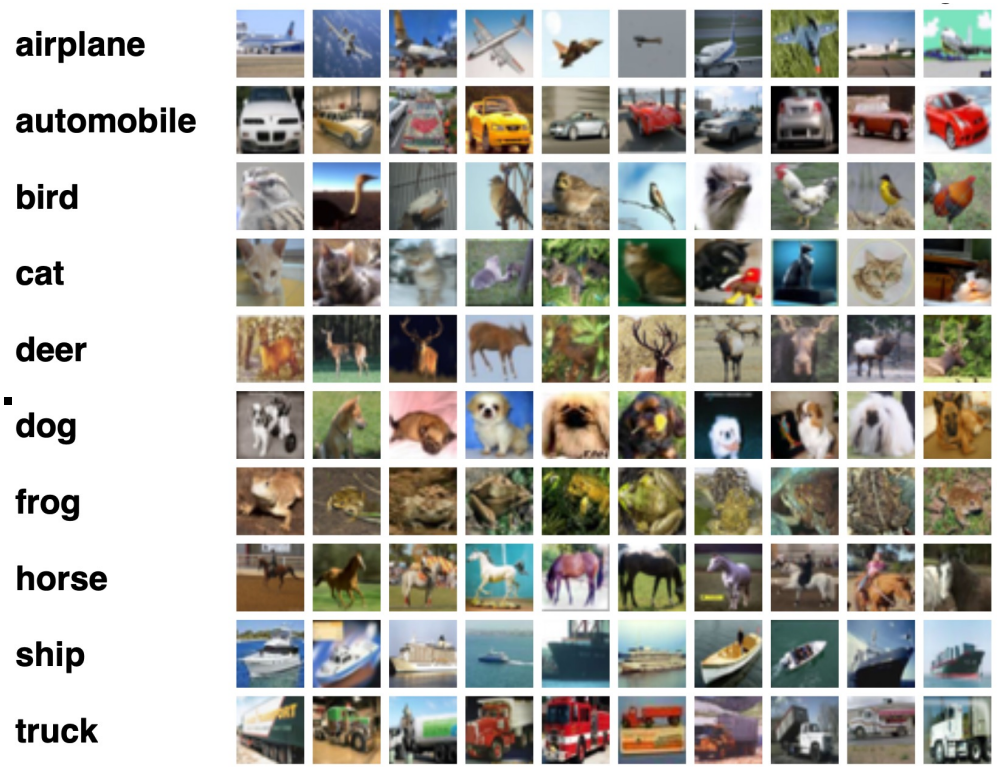
Live demo http://scs.ryerson.ca/~aharley/vis/fc/

http://yann.lecun.com/exdb/mnist/

https://www.kaggle.com/oddrationale/mnist-in-csv/home

Datasets: CIFAR

- CIFAR-10
  - 32x32 colour images
  - 60000 in 10 classes
  - 6000 images per class.
  - 50000 training images
  - 10000 test images



- CIFAR-100
  - 100 classes with 600 each
  - 20 superclasses

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

IMAGENET

Datasets: Imagenet Challenge

- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC from 2010)
- Goal to push the state of the art in computer vision (human vision approximately 95– 96%).
- 200 possible classes x 450,000 images
- Challenges:
  - Object localization for 1000 categories.
  - Object detection for 200 fully labeled categories.
  - Object detection from video for 30 fully labeled categories

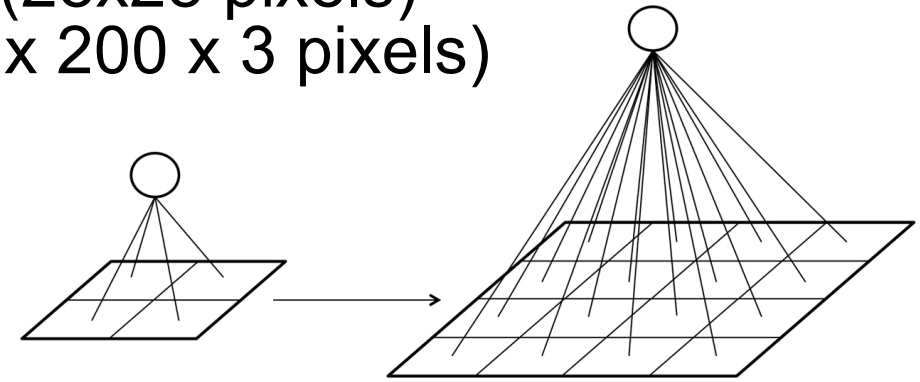
http://www.image-net.org

CNN: Convolutional Neural Networks

CNN for Computer Vision

- From human-defined features (e.g. SIFT)
- To learned features (filters)
  - Same as hierarchical, deep brain visual cortex

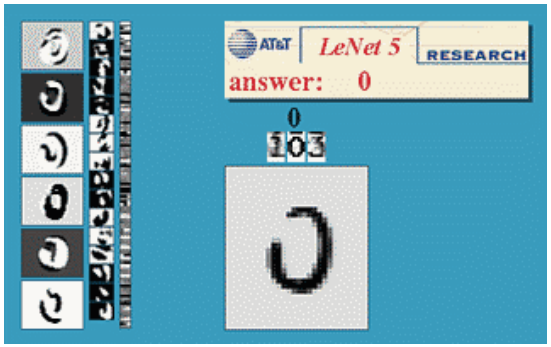
- Scale-up and intractability
  - From MIST 784 weights (28x28 pixels) to 120,000 weights (200 x 200 x 3 pixels)
  - Reduced connectivity
  - Repeated filters
  - Pooling units

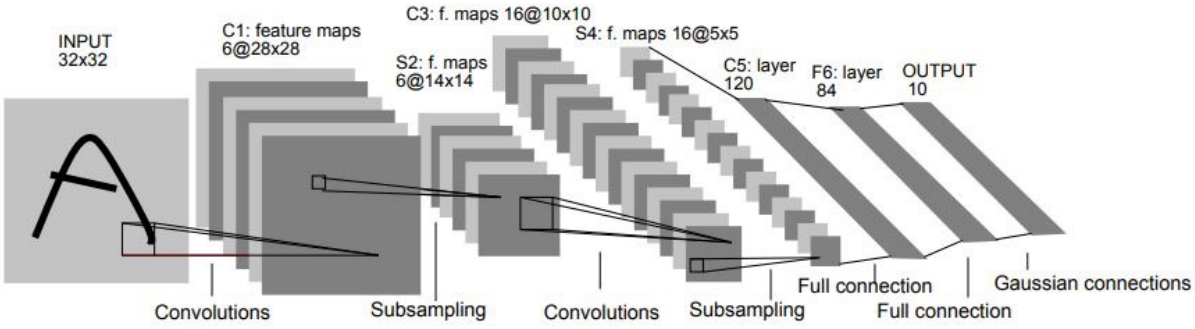


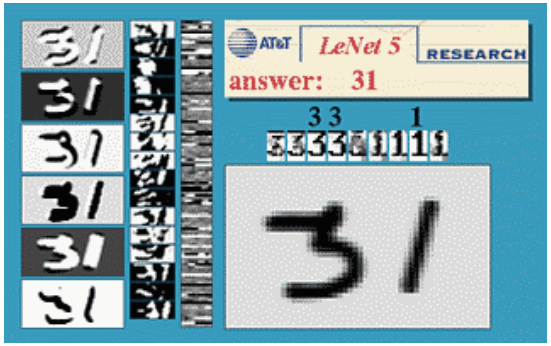
- Revolution in 2012 AlexNet (1998 LeNet-5)

LeNet

- First convolution network (1998)
- OCR for numbers and letters
- MNIST dataset (M-NIST)





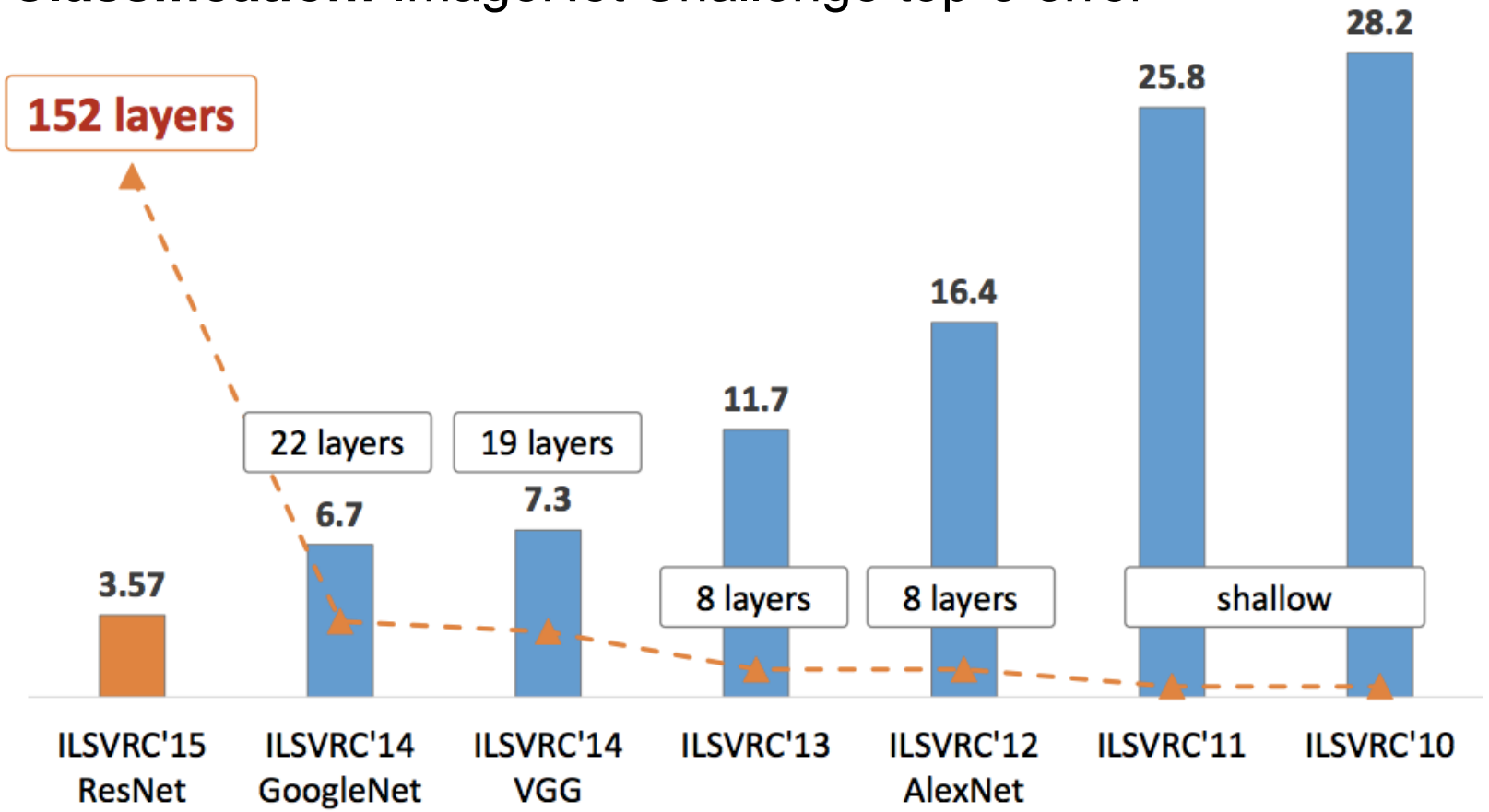


LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324 yann.lecun.com/exdb/publis/pdf/lecun-

IMAGENET

ImageNet Challenge

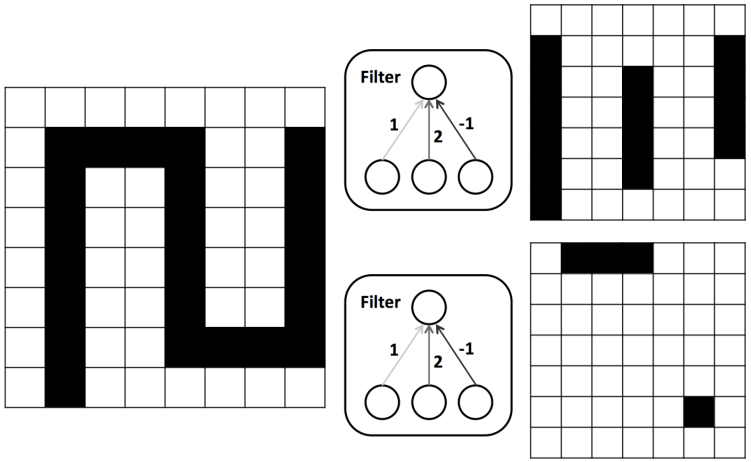
Classification: ImageNet Challenge top-5 error

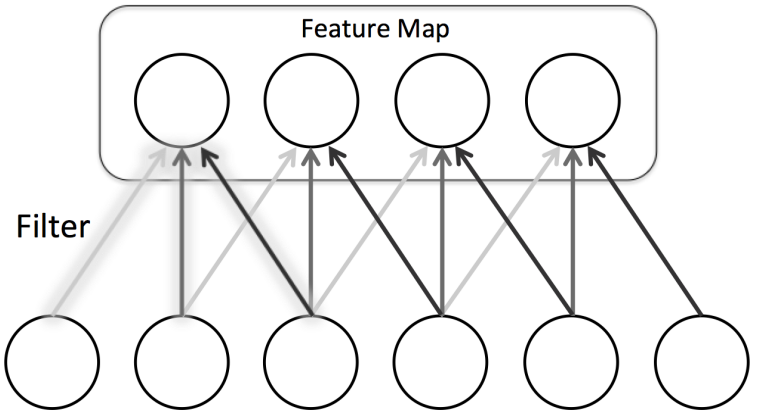


slazebni.cs.illinois.edu

Feature Map

- Simple neurons use **local receptive fields** to **filter** elementary features (e.g. edges, end-points, corners…)
  - A neuron in the feature map activated if filter detects appropriate feature at the corresponding position in the previous layer





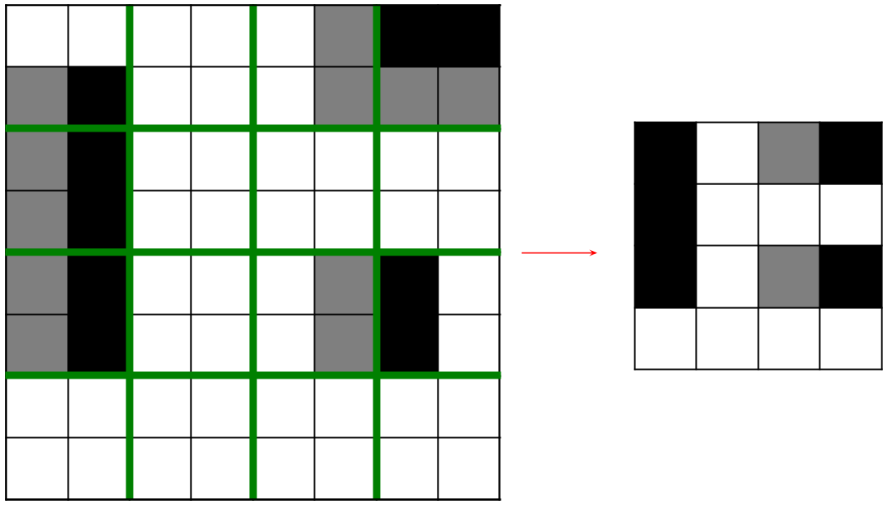
Keras Code example

- Convolution layer

```
from keras.layers.convolutional import Conv2D
from keras.layers.core import Activation

model.add(Conv2D(20, kernel_size=5, padding="same", input_shape=input_shape))
model.add(Activation("relu"))
```

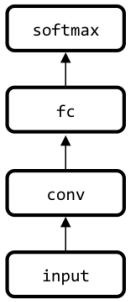
Pooling (sub-sampling)

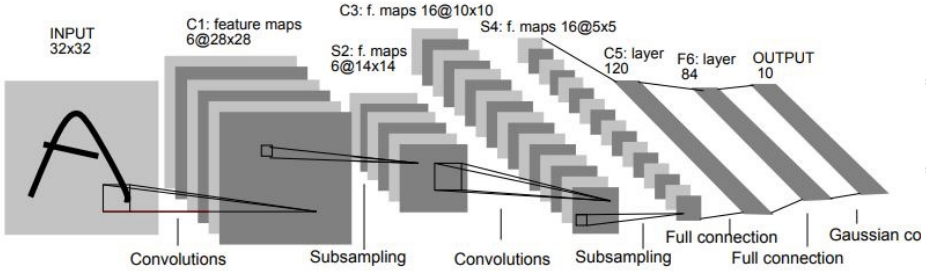
- To **reduce spatial resolution** of feature maps
- **Locally invariant**: Reduces the sensitivity of the output to shifts and distortions
- **Max** pooling
  - take max value from the receptive field values



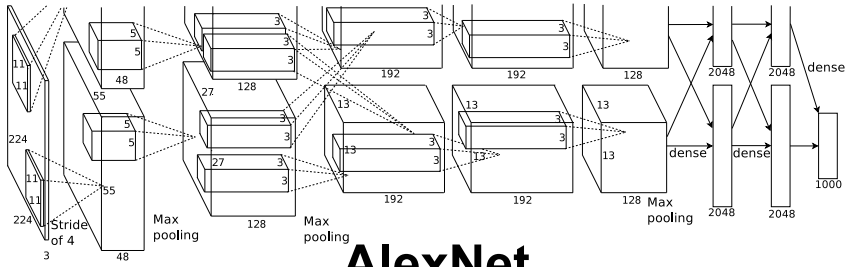
CNN topologies

- Minimal topology
  - Input → convolution → fully connected → softmax
- Convolution-Pooling module(s)
  - Input → (convolution→pooling) →
    fully connected → softmax

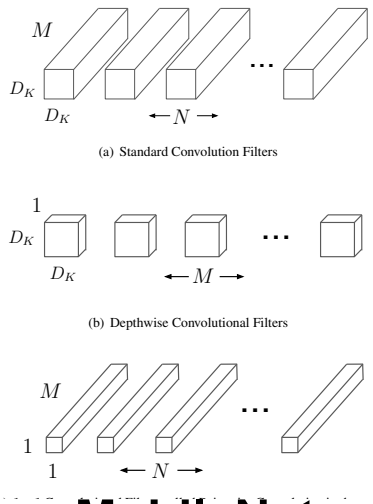




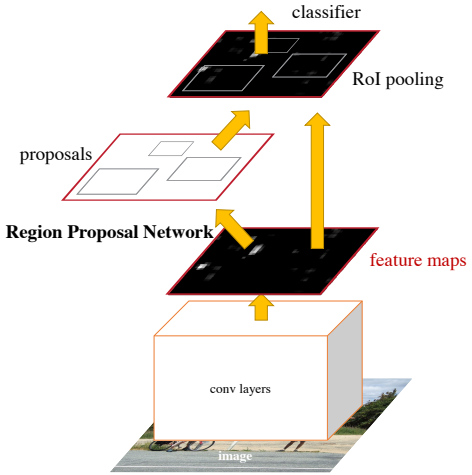
LeNet-5



AlexNet

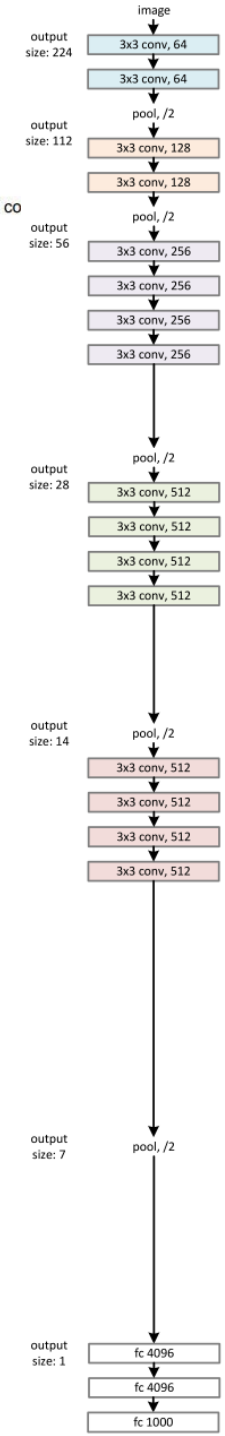


MobileNet

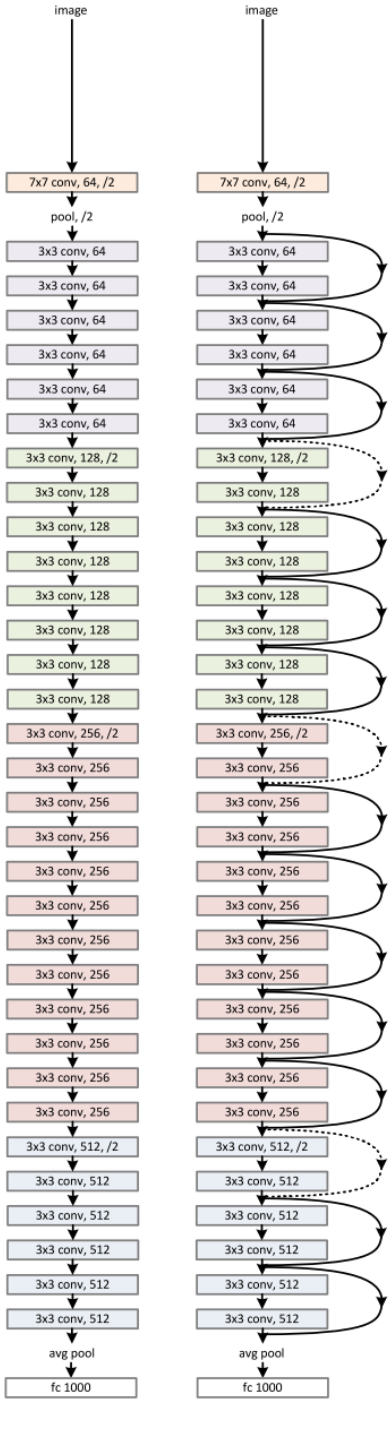


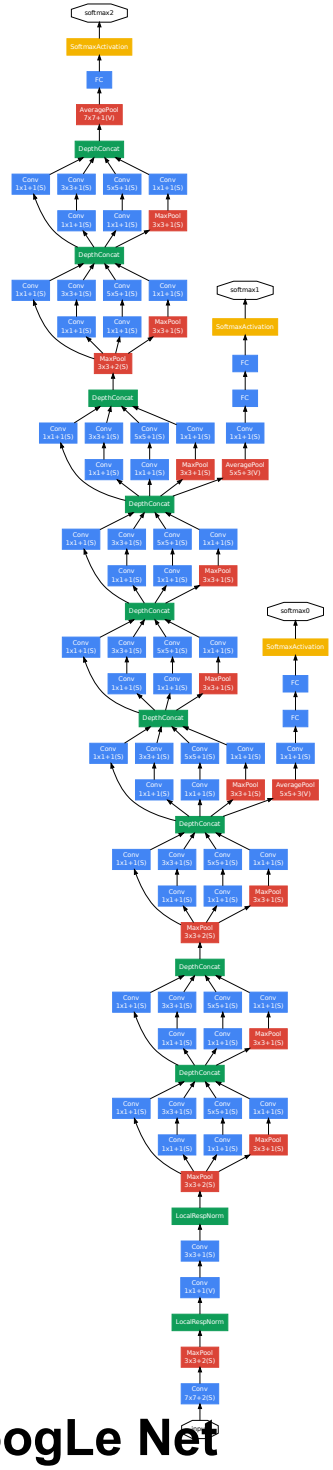
Faster RCNN

VGGNet



ResNets



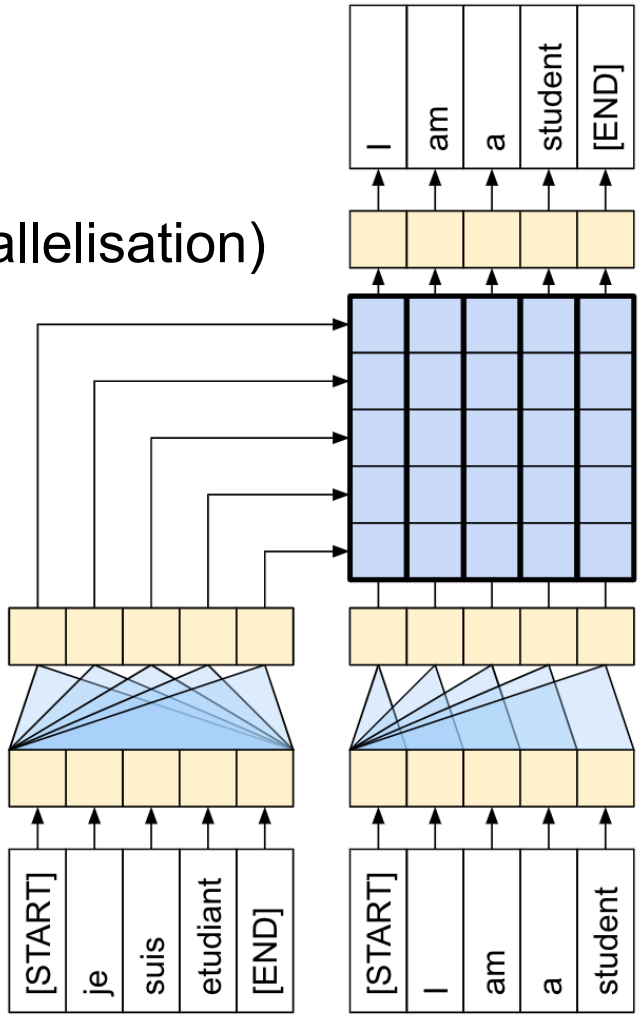


GoogLe Net

Transformers

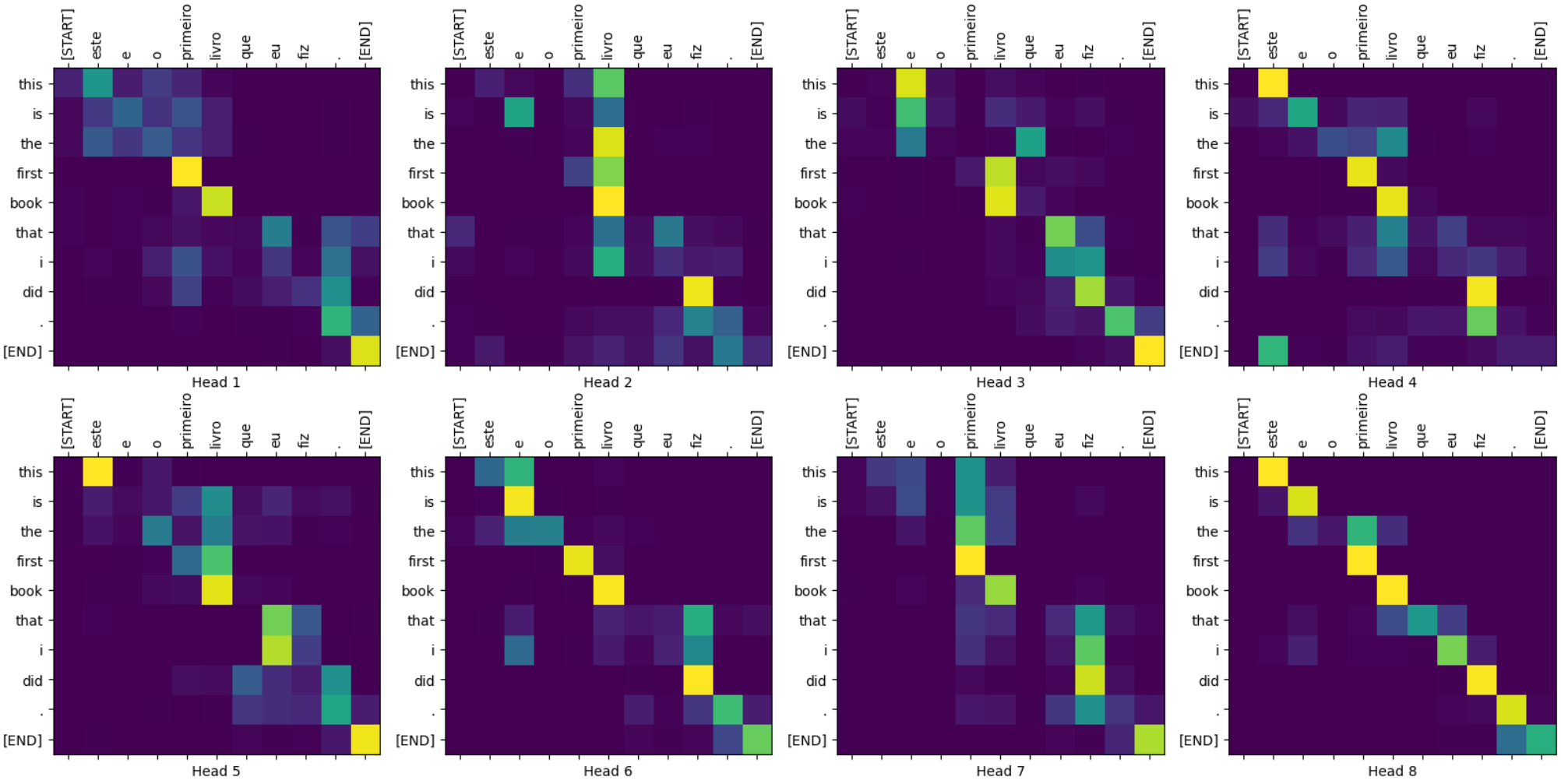
Transformers

- Attention mechanism
  - replaced recurrence with attention (parallelisation)
  - long-range contexts and dependencies



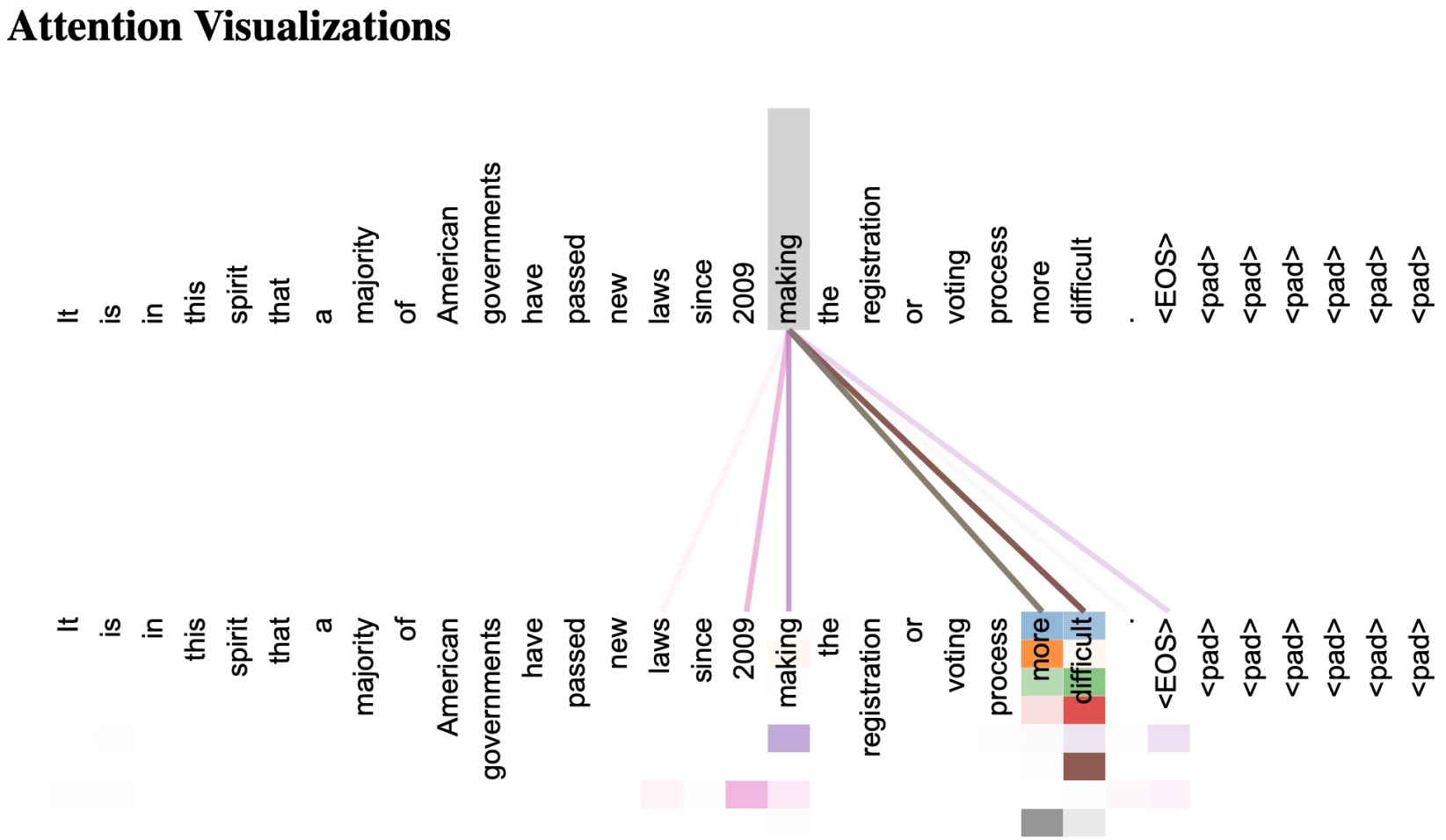
Keras Tutorial Translation Transformer: https://www.tensorflow.org/text/tutorials/transformer

Transformers



Keras Tutorial Translation Transformer: https://www.tensorflow.org/text/tutorials/transformer

Transformers

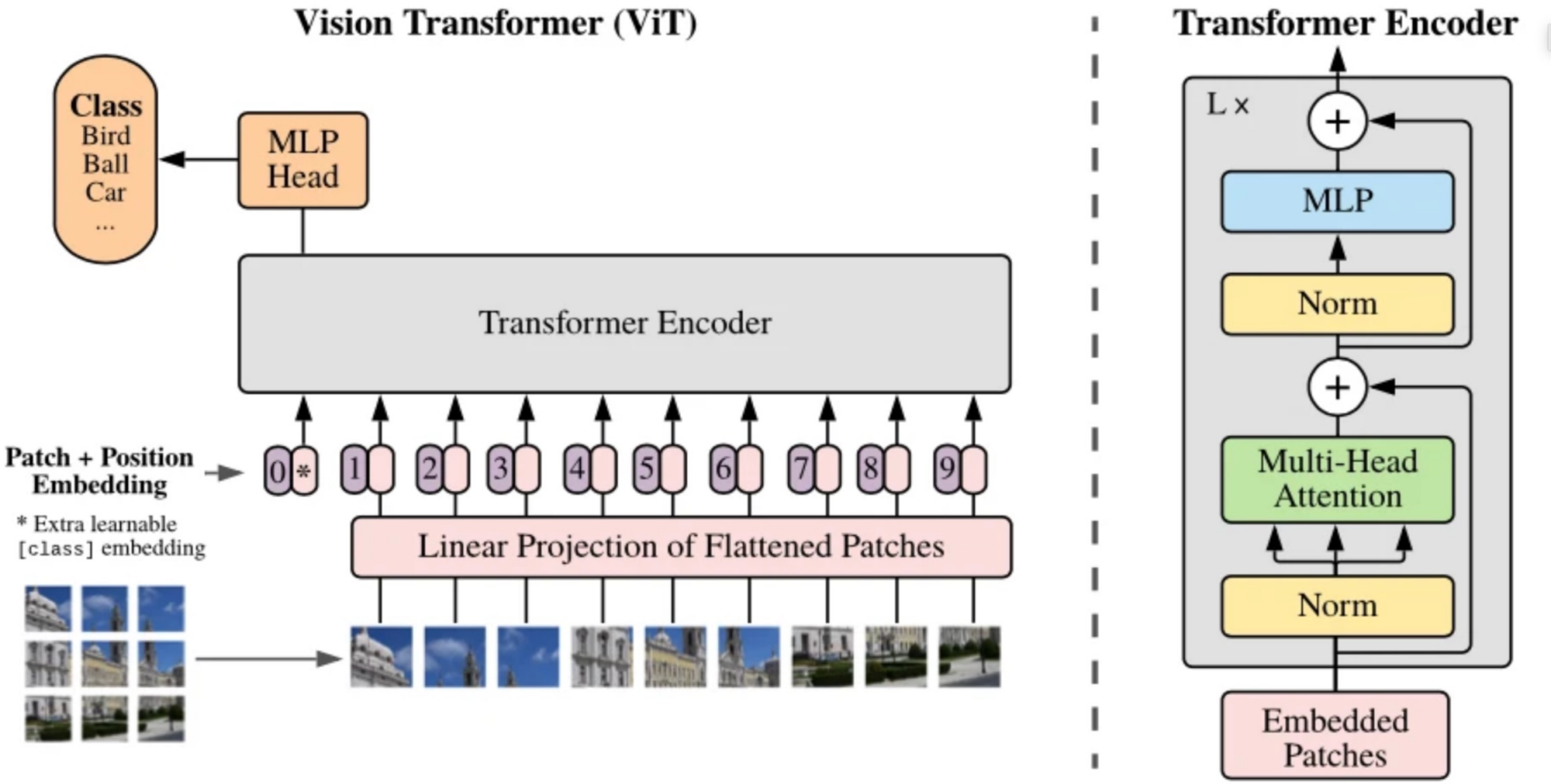


Vaswani et a. (2017), Attention Is All You Need. https://arxiv.org/abs/1706.03762

Transformers

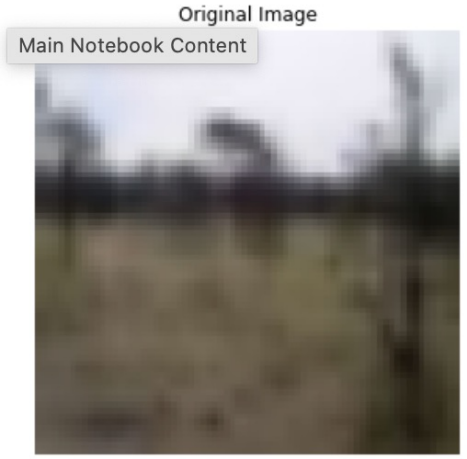
- Attention mechanism
  - replaced recurrence with attention (parallelisation)
  - long-range contexts and dependencies
- Process sequential data
  - Language as primary sequential problem
    - From BERT to GPT
    - Translation, summarisation, multiple generative tasks (ChatGPT)
  - Application computer vision: ViT for images
  - Multimodal transformers (e.g. image/video captioning)

ViT: Visual Transformers

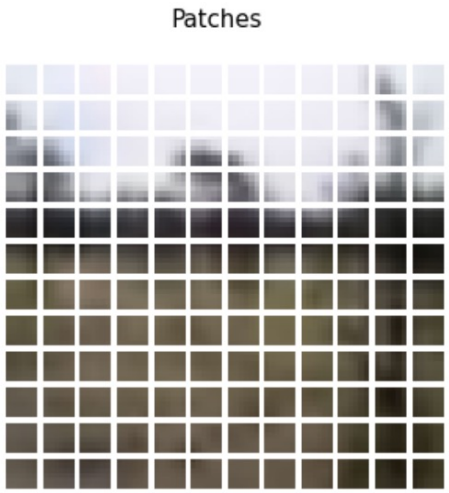


Keras Tutorial ViT https://www.kaggle.com/code/utkarshsaxenadn/vit-vision-transformer-in-keras-tensorflow/notebook

ViT: Visual Transformers



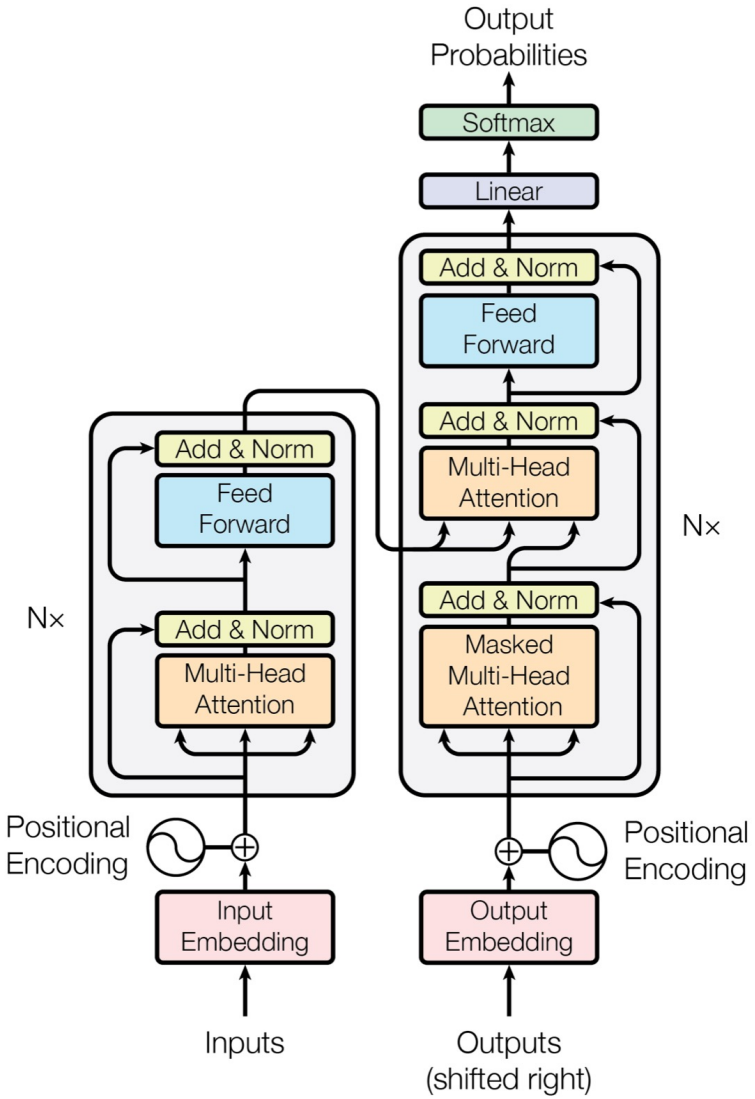
- Alternative to CNN
- Input layers
  - Image as patches with position encoding
  - Augmentation before patches
- Transformer/Attention layers
  - Multi-Head Attention x patches relationship
  - Normalisation/residual layers
- Output labels with MLP
- Celebrity: Visual Transformer, DINOv2

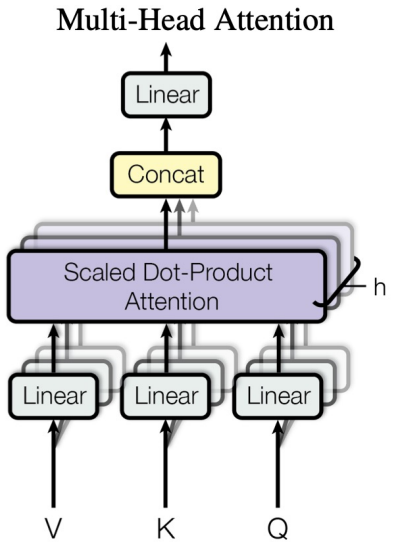


Dosovitskiy et al. (2000). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. https://arxiv.org/abs/2010.11929

Keras Tutorial ViT https://www.kaggle.com/code/utkarshsaxenadn/vit-vision-transformer-in-keras-tensorflow/notebook

Transformers for NLP





Vaswani et a. (2017), Attention Is All You Need. https://arxiv.org/abs/1706.03762

https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Transformers for NLP

- For multiple language tasks
  - Translation, parsing/tagging…
  - Large Language Models (LLM)
    - Transformer (2017), GPT1 and BERT (2018), GPT3 (2020)
    - AI for people: ChatGPT (2023)
    - Tokenisation, Reinforcement learning from human feedback (RLHF), Prompt engineering…
  - Ethics of AI since widespread LLM impact

Keras Tutorial Translation Transformer: https://www.tensorflow.org/text/tutorials/transformer

Summary

- MLP: Multi-Layer Perception
- Learning algorithms and datasets
- CNN: Convolutional Neural Networks
  - Convolution filters, pooling, deep structure
- Transformers
  - Attention, ViT, LLM
- **Reading**
  - Sünderhauf et al. IJRR (2018) DNN for robotics
    (e.g. robotics and DNN contextualisation / report!)
  - Tutorials (labs + optional on Transformers)