SimCenter NHERI
Center for Computational Modeling and Simulation

2021 Programming Bootcamp

Neural Network Architectures

Yunhui Guo

University of California, Berkeley

NSF award: CMMI 1612843

Outline

Part 1 Multi-Layer Perceptron

Machine learning

Multi-layer perceptron

Backpropagation

Evaluation

Demo

Part 2 Convolutional Neural Networks

Why do we need convolutional neural networks?

Convolution, normalization, pooling

Convolutional neural networks as feature extractor

Demo

Outline

Part 3 Attention Networks

Attention in human visual system

Introduce attention into convolutional neural networks

Residual attention network

Demo

Part 4 Transformers

Transformers for image classification

Transformer encoder

Self-attention layer

Inspecting transformer

Demo

Part 1
Multi-Layer Perceptron

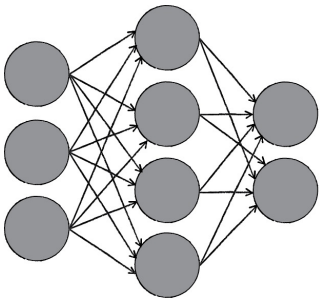
Machine Learning

Task: classify a given image

2

Data

Model



Algorithm

More Applications of Machine Learning

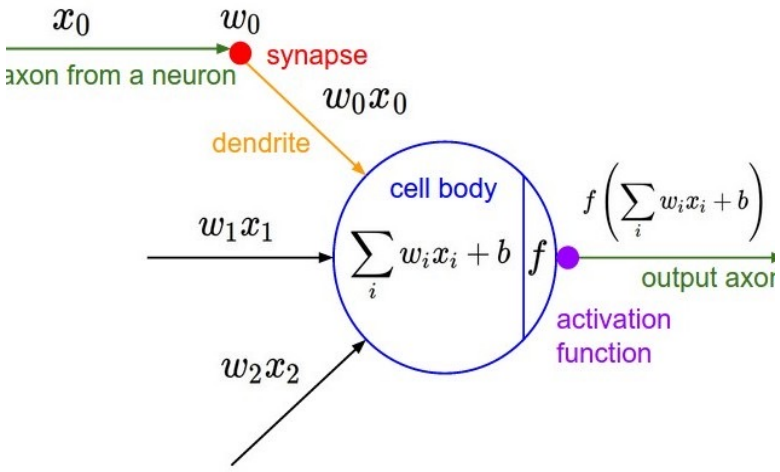
Autonomous Driving

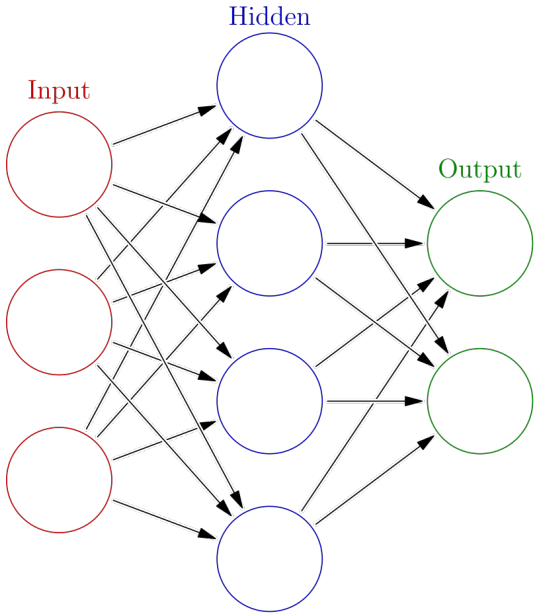
Playing Board Games

Playing Dota2

Neural Networks

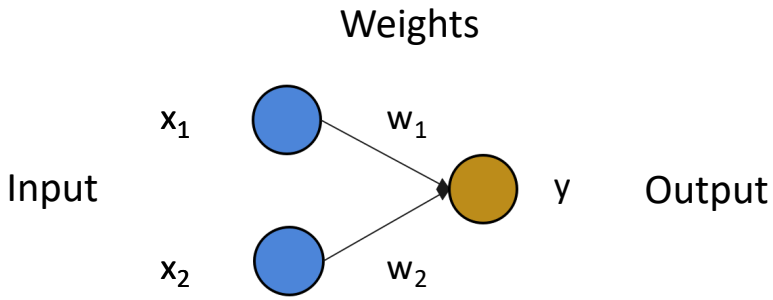
- Neural networks consist of a collection of nodes which model biological neurons in human brain.
- Neural networks have one input layer, one or multiple hidden layers and one output layer





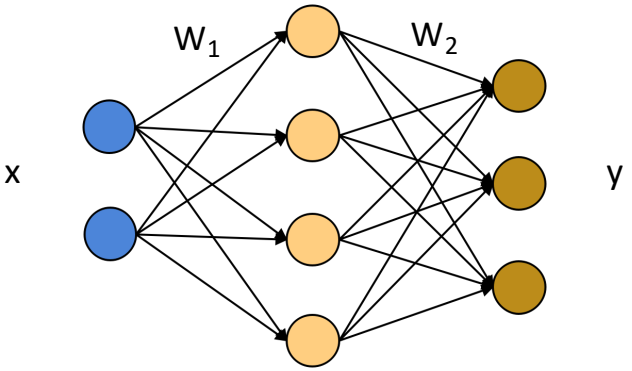
Multi-Layer Perceptron

Single-Layer Perceptron



$y = x_1 \times w_1 + x_2 \times w_2 + bias$

Multi-Layer Perceptron



$y = W_2\,(\sigma(W_1 x + bias)) + bias$

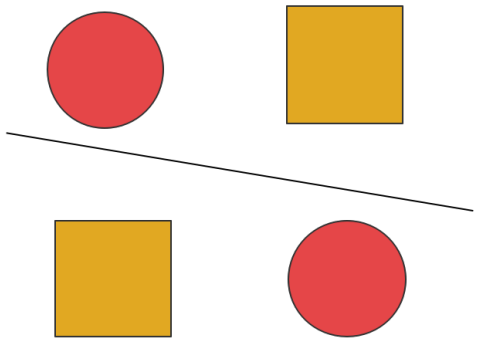
$\sigma(x)$ is an activation function

Why Do We Need Activation Functions?

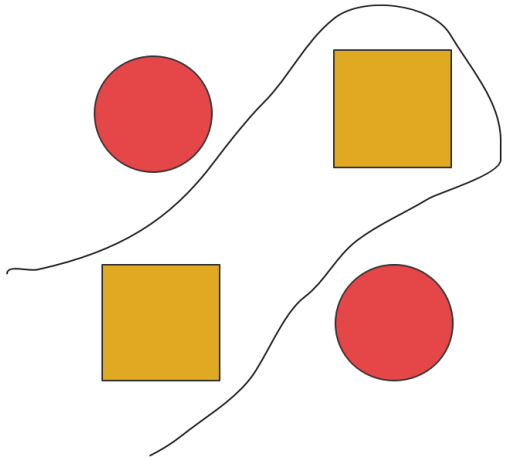
Multi-layer peceptron is a linear function without activation functions:

- The classification plane is linear which is not sufficient for complex tasks.

Linear classification plane



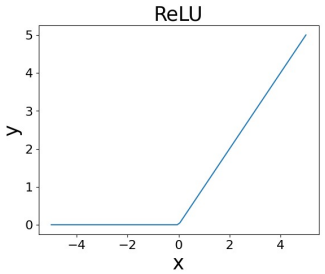
Nonlinear classification plane



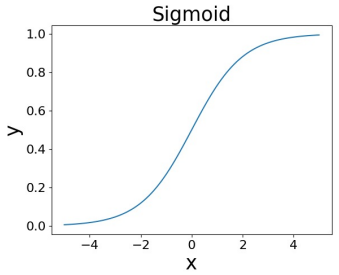
Activation Function

Activation function introduces non-linearity in the network.

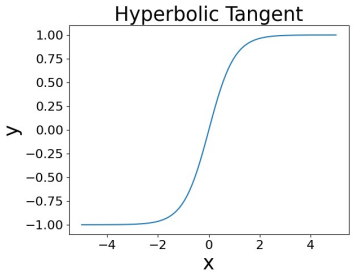
- Rectified Linear Unit (ReLU): $\sigma(x) = \max(0, x)$



- Sigmoid: $\sigma(x) = \dfrac{1}{1 + e^{-x}}$



- Hyperbolic Tangent: $\sigma(x) = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$



Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

Multi-Layer Perceptron: Training

- Before training, the weights are randomly initialized. The training adjusts the weights to make correct predictions.

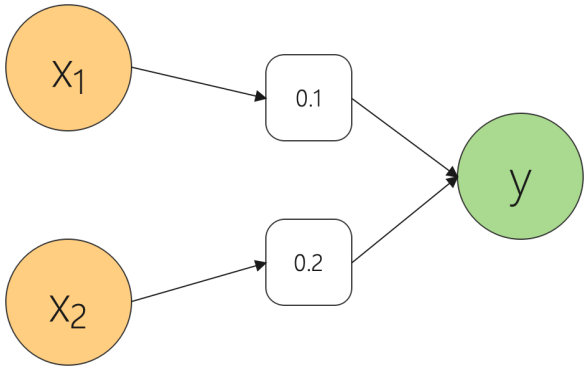
- We need a loss function *L* and a training set to train the network

- Training is done via gradient descent (find the minima of the loss):

  - $W = W - \epsilon \frac{\partial L}{\partial W}$ ϵ is the learning rate

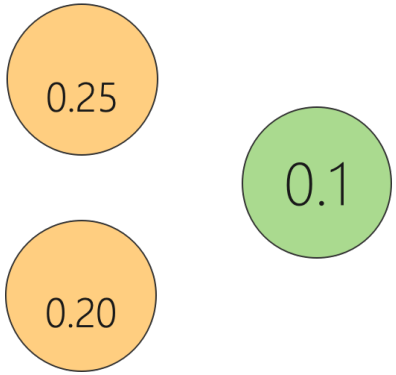
Training A Single-Layer Perceptron

Initialization



$$y = 0.1x_1 + 0.2x_2$$

Training data



$$\hat{y} = 0.065 = 0.1 * 0.25 + 0.2 * 0.2$$

Loss function

$$L(y, \hat{y}) = (y - \hat{y})^2$$

Training A Single-Layer Perceptron

The gradient of L with respect to w_1

$$\frac{\partial L}{\partial w_1} = -2 * (y - w_1 x_1 - w_2 x_2) x_1 = -0.0175$$

The gradient of L with respect to w_2

$$\frac{\partial L}{\partial w_2} = -2 * (y - w_1 x_1 - w_2 x_2) x_2 = -0.014$$

Update weights

$$w_1 = w_1 - \frac{\partial L}{\partial w_1} = 0.1175 \qquad w_2 = w_2 - \frac{\partial L}{\partial w_2} = 0.2014$$

Prediction is more accurate

$$\hat{y} = 0.0696 = 0.1175 * 0.25 + 0.2014 * 0.2$$

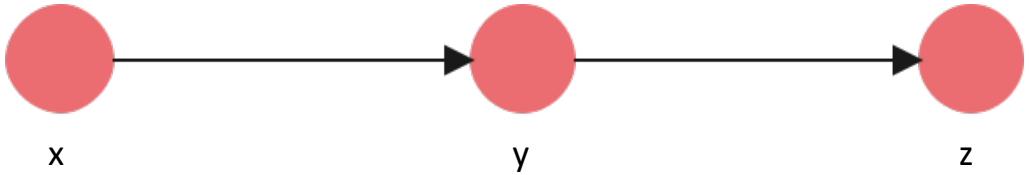
Back-propagation

- Back-propagation computes the gradients of the weights with respect to a loss function

- Chain rule in calculus:
  - Suppose we have a function L(x) = f(g(x)), then

$$L'(x) = f'(g(x))g'(x)$$

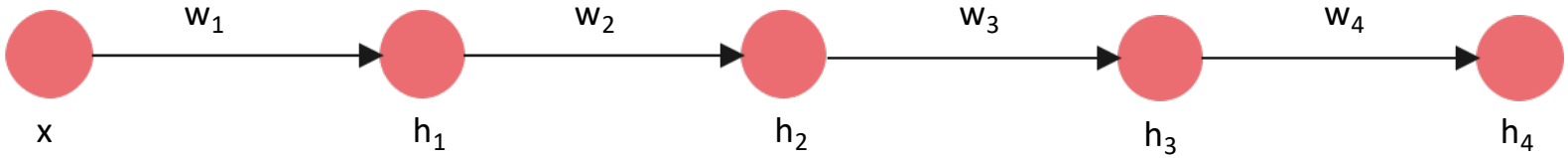
- z is a fuction of y and y is a function of x:



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

Back-propagation

- A toy multi-layer perceptron:



- The loss L is a function of h_4
- The relation between h_{i+1} and h_i:

$$h_{i+1} = \sigma(W_{i+1}h_i + \text{bias}) \qquad (1)$$

By chain rule:

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial h_i}\frac{\partial h_i}{\partial w_i}$$

$$\frac{\partial L}{\partial h_i} = \frac{\partial L}{\partial h_{i+1}}\frac{\partial h_{i+1}}{\partial h_i}$$

Evaluation

- We evaluate the performance of the trained model on a given test set

- The network ends with the softmax function. The index which achieves the maximum value is the prediction,

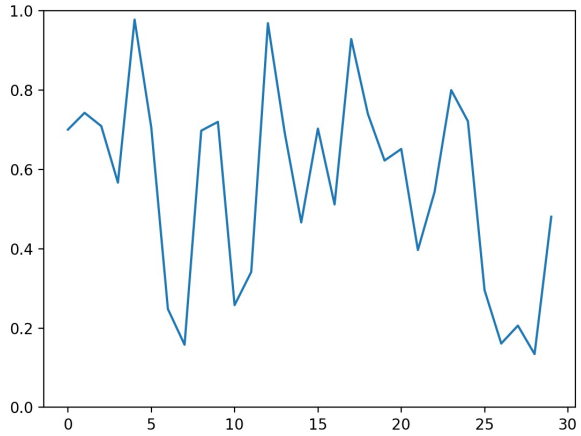
$$\text{Softmax}(\mathbf{Z})_i = \frac{e^{Z_i}}{\sum_{j=1}^{K} e^{Z_j}}, \qquad \text{for} \quad i = 1, ..., K \quad \text{and} \quad \mathbf{Z} = \{Z_1, ..., Z_K\} \quad \text{is the unnormalized logits}$$

- Test Accuracy:

$$\frac{\text{Number of correct predictions}}{\text{Number of test examples}}$$

MLPs are Universal Function Approximators

- MLP can approximate any well-behaved function.



$$\sup_{x \in K} ||f(x) - MLP_{\epsilon}(x)|| < \epsilon$$

Question

- Which one is false?

  a. If the learning rate is too small, the training willl progress slowly.

  b. If the learning rate is too large, the loss function might diverge.

  c. The training data can be used only once during training.

  d. MLPs with activation function can approximate nonlinear classification plane.

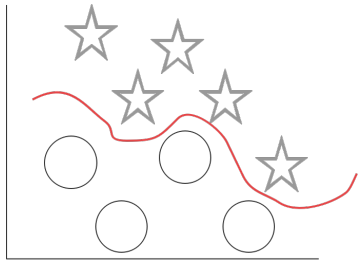
Demos

Will do demos in Jupyter notebooks

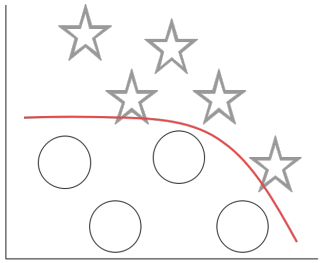
Part 2
Convolutional Neural Networks

Overfitting in Machine Learning

- Overfitting:
  - The model only fits against the training data

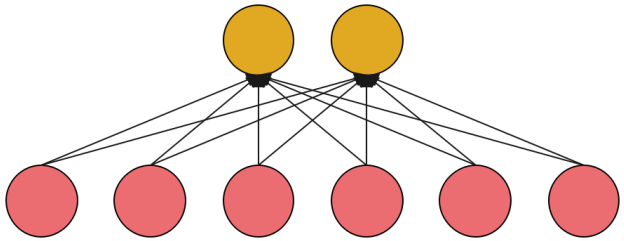


- What we really want:
  - The model can perform well on the unseen test data

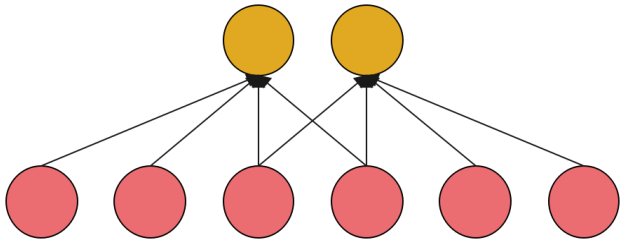


Why Do We Need Convolutional Neural Networks (CNNs) ?

- Multi-Layer Peceptrons are "fully-connected" :
  - Prone to overfitting

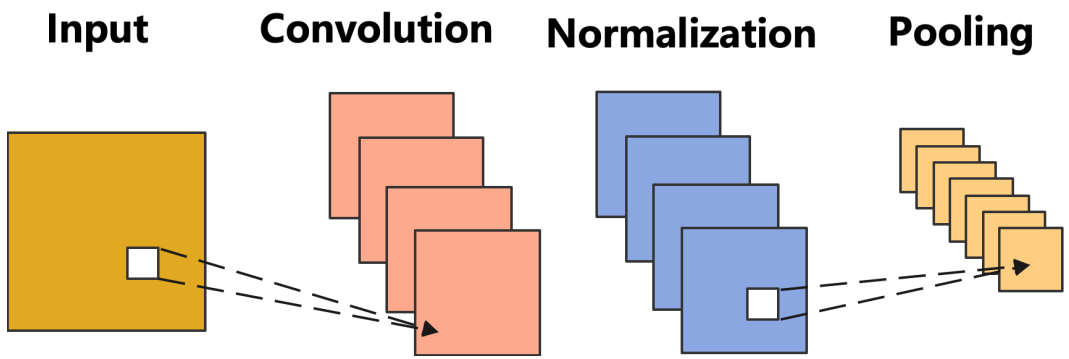


- In convolutional neural networks, each neuron only connects a restricted region of the input.



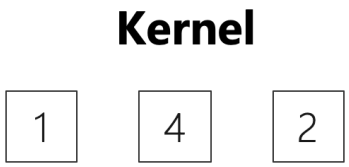
Convolutional Neural Networks (CNNs)

- Convolutional Layers
- Pooling Layers
- Normalization Layers

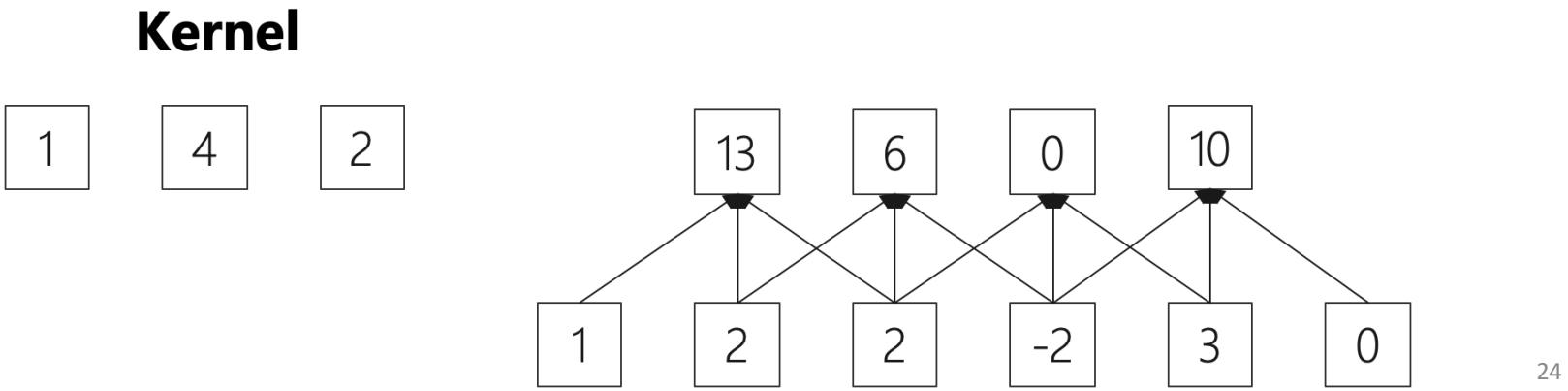


Convolutional Layers

- The convolutional layer consists of a set of learnable kernels



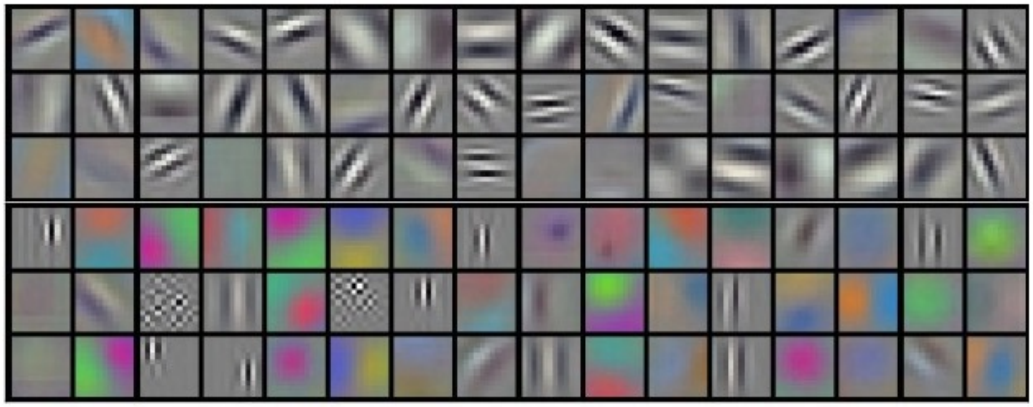
- During the forward pass, we slide each kernel across the input and compute dot products between the kernel and the input at each position.
- A one-dimensional example:



Convolutional Layers

- Parameter sharing: one convolutional filter can be applied to different positions of the input to detect the same pattern.

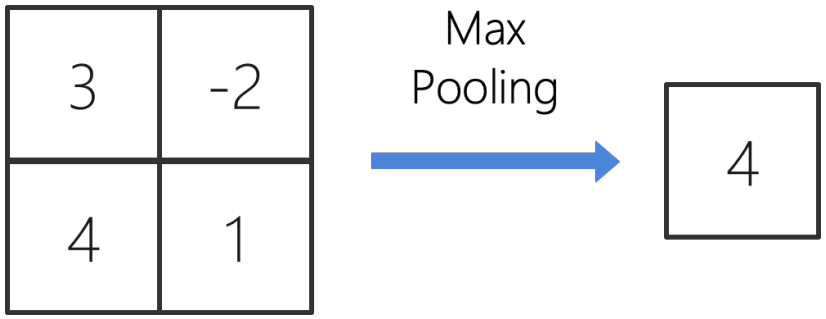
- Example filters learned by AlexNet (Krizhevsky et al).



Pooling Layers

- Reduce the spatial size of the representation to reduce the amount of parameters and control overfitting

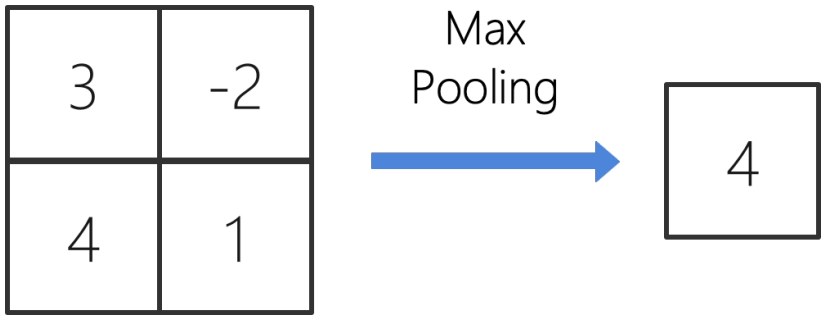
- Max pooling operation:
  - Pooling size is 2 X 2



Pooling Layers

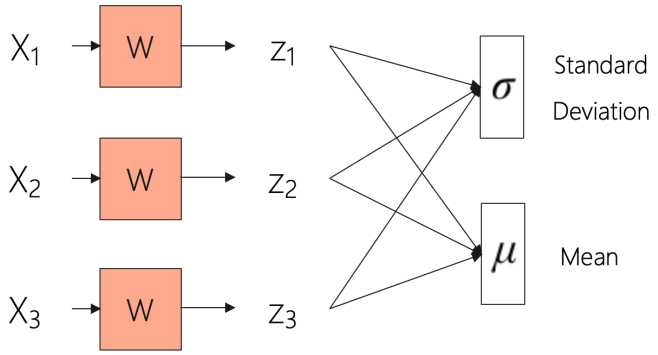
- Pooling layer can also reduce the sensitivity of the convolutional layer to different locations:
  - Invariant to translation.

- Max pooling operation:
  - Pooling size is 2 X 2



Normalization Layers

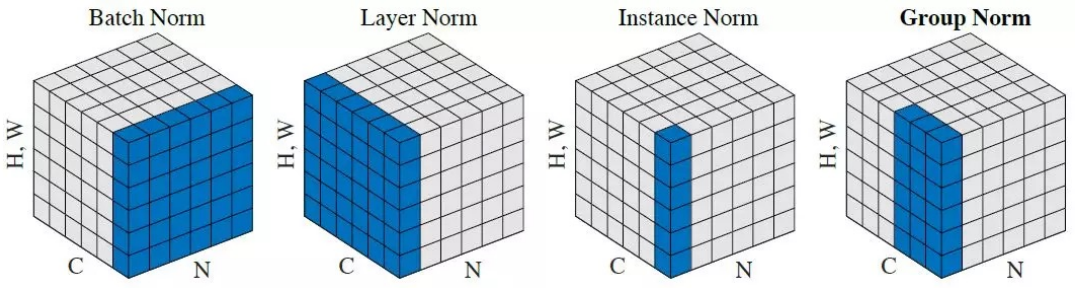
- Batch normalization:
  - more stable training by re-centering and re-scaling the inputs.



$$\hat{z}_i = \frac{z_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \qquad (1)$$

$$y_i = \mathrm{BN}_{\gamma,\beta}(z_i) = \gamma \hat{z}_i + \beta$$

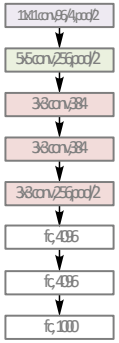
- Different normalization methods



Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

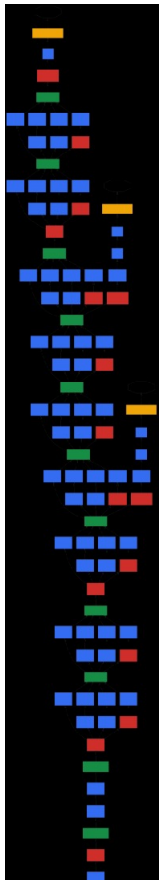
Case Studies

AlexNet, 8 layers (ILSVRC 2012)



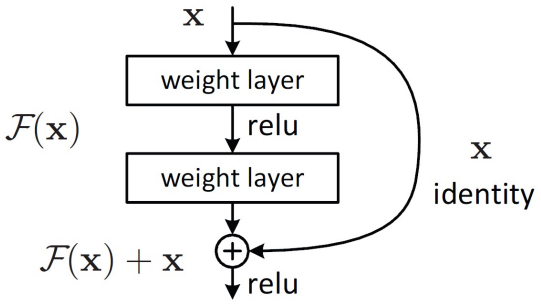
VGG, 19 layers (ILSVRC 2014)

GoogLeNet, 22 layers (ILSVRC 2014)



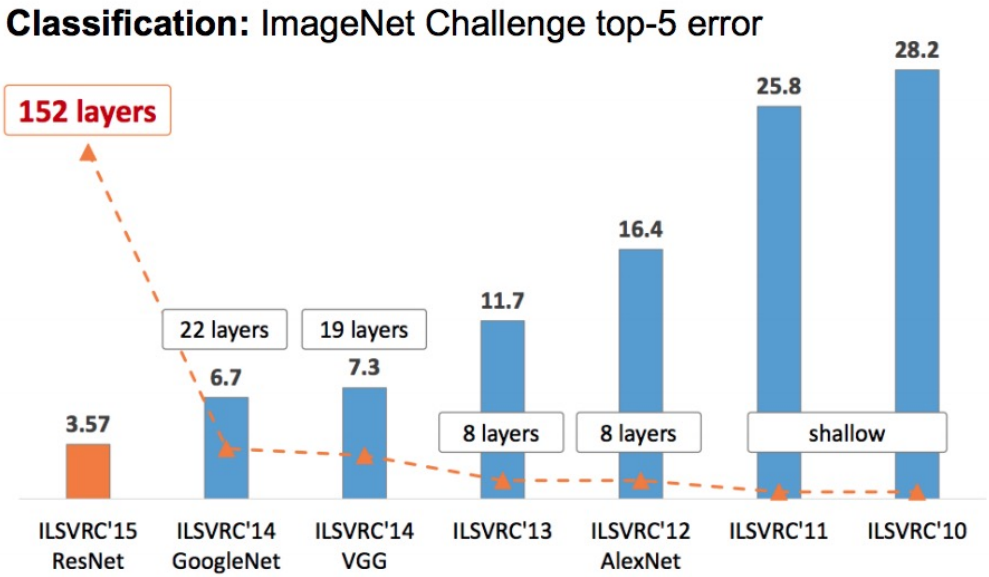
ResNet, 152 layers (ILSVRC 2015)

Residual connection

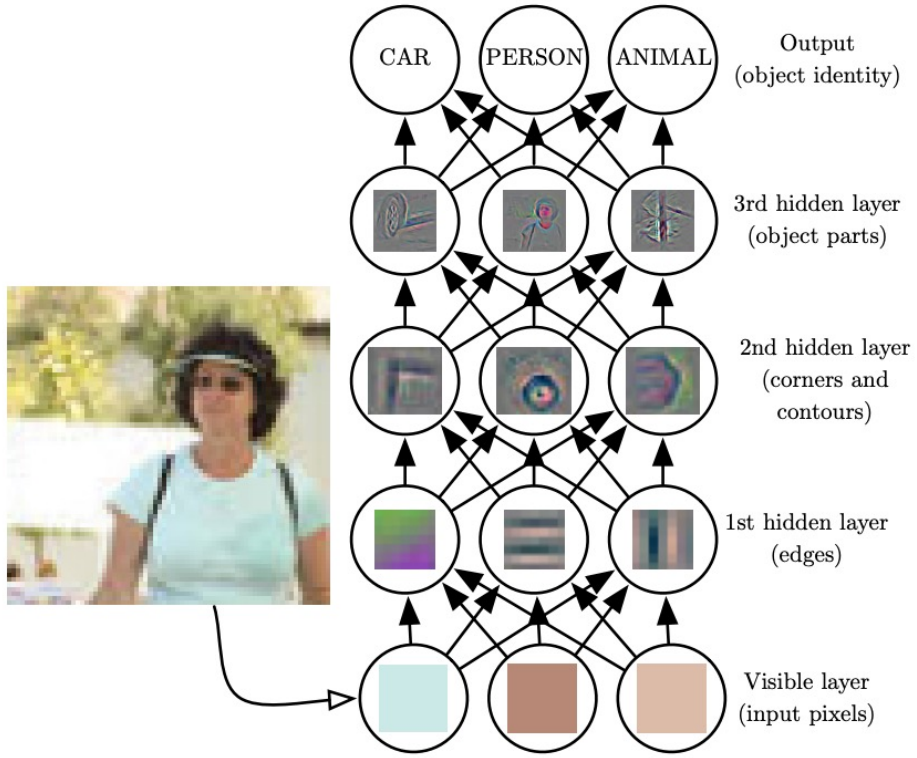


Progress on ImageNet

- ImageNet dataset:
  - A large-scale dataset for image classification
  - 1,000 categories
  - 1,281,167 images for training
  - 50,000 images for validation



What is Learnt by Convolutional Neural Networks?



Question

- Which one is false?

  a. Different kernels in CNN can detect different patterns.

  b. Deeper networks always achieve better accuracy than shallower networks.

  c. Convolution can reduce the spatial size of the input.

  d. Initial layers of a CNN learn very generic features.

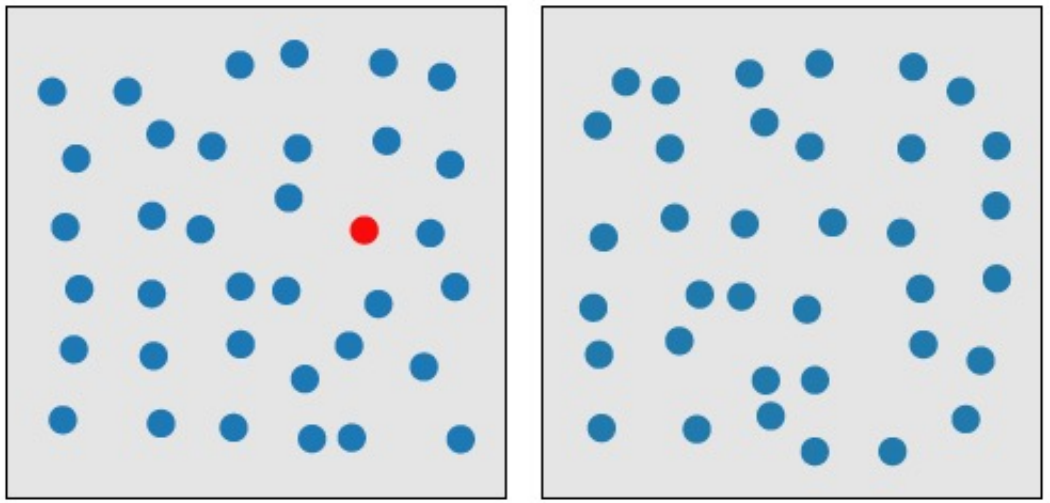
Demos

Will do demos in Jupyter notebooks

Part 3
Attention Networks

Attention in Human Visual System

- What we see can depend critically on where attention is focused.



Why Do We Need Attention Networks?

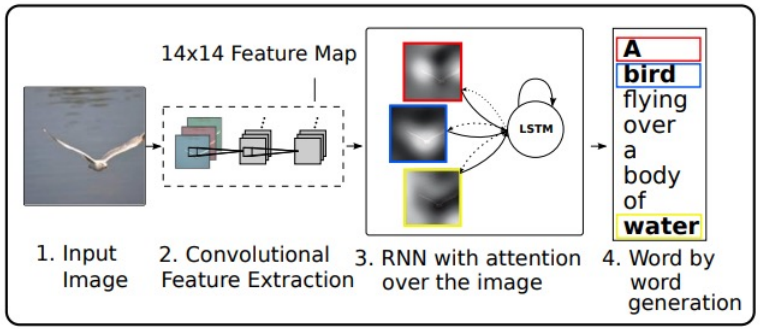
- Convolutional neural networks are powelful but they lack the ability to accurately locate the objects in the input image.

- Consider the task of generating captions given an image,

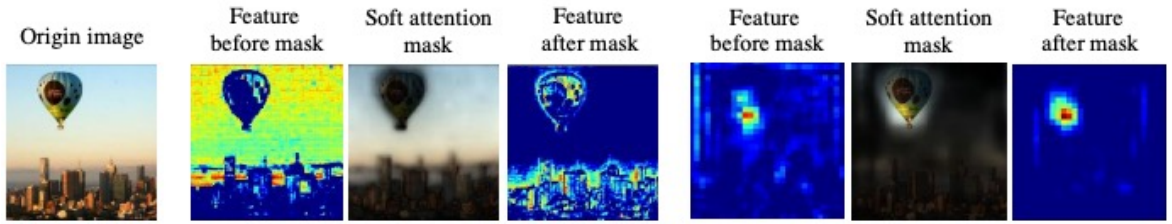
Two dogs running in a grass field

Introduce Attention into Convolutional Neural Networks

- Attention mechanism is applied to image caption generation.



- Attention is used to refine the convolutional features



Residual Attention Network

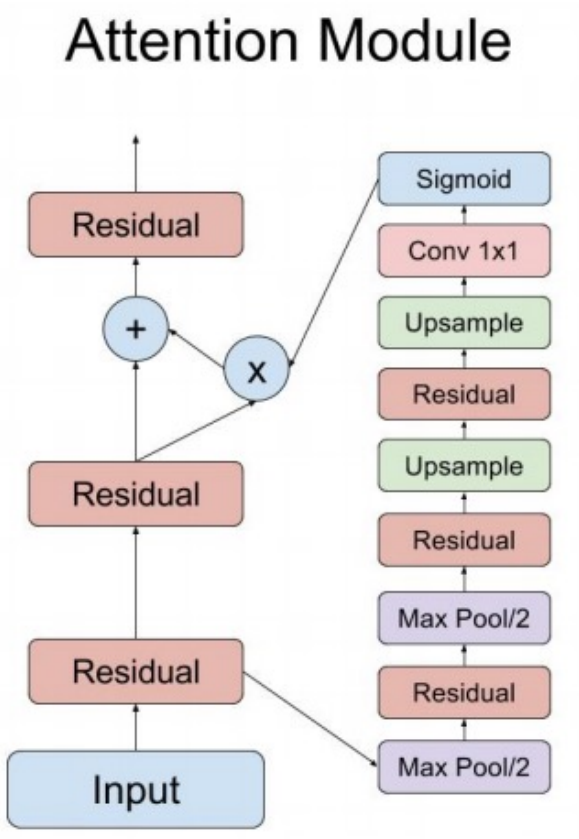
- The network has several attention modules.
- Each attention module has a truck branch *F* and a mask branch *M.*
- The output of the attention module is computed as,

$$H(x) = (1 + M(x)) \cdot F(x)$$

  - The output of the mask branch ranges from [0, 1] and works as feature selectors which enhance good features and suppress noises from trunk features

Bottom-up Top-down Structure in the Mask Branch

- Collects global information of the whole image
  - Downsampling the input via max pooling
  - Incorporate residual connection
  - Upsample to the input size via bilinear interpolation
  - Compute the sigmoid activation

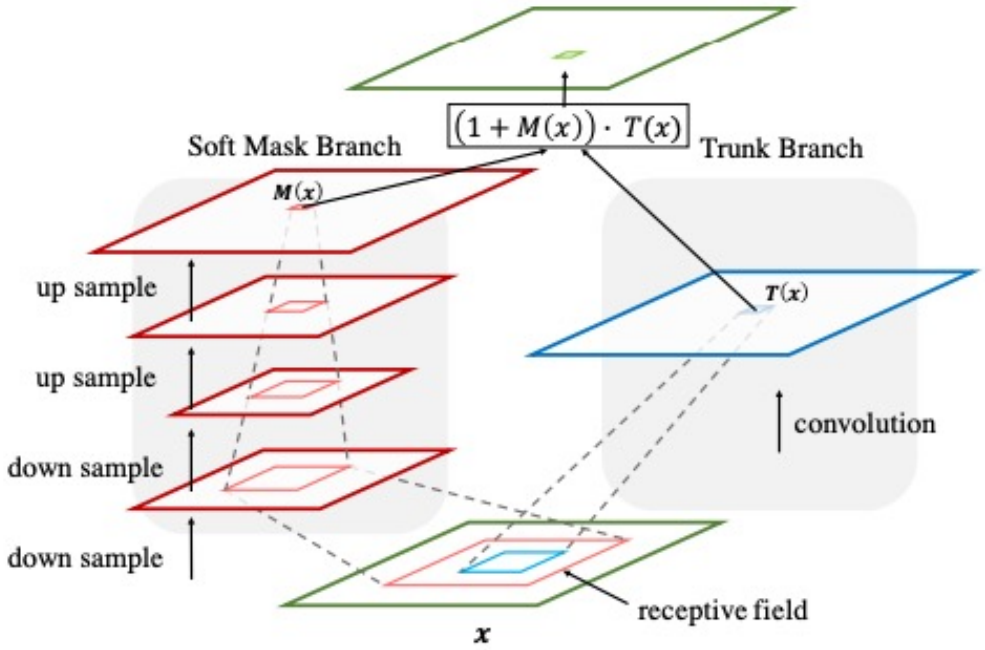


Different Attention Functions

- Mixed attention $$f_1(x_{i,c}) = \frac{1}{1+\exp(-x_{i,c})}$$
- Channel attention $$f_2(x_{i,c}) = \frac{x_{i,c}}{||x_i||}$$
- Spatial attention $$f_3(x_{i,c}) = \frac{1}{1+\exp(-(x_{i,c}-mean_c)/std_c)}$$

Overall Architecture

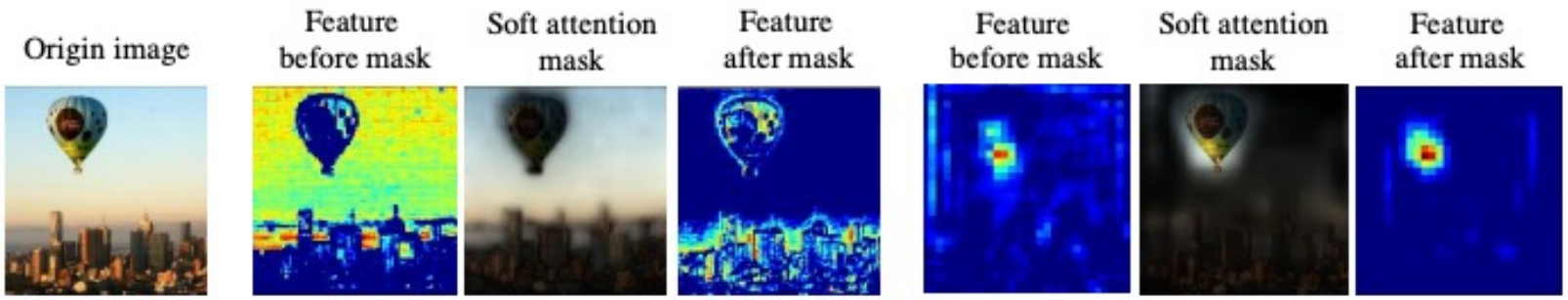
- The mask branch serves as feature selectors which enhance good features and suppress noises from trunk features similar to the attention mechanism in human visual system.



Performance of Attention Networks

- Compared with NAL (without attention), attention networks achieves better results on CIFAR-10.

Network	ARL (Top-1 err. %)	NAL (Top-1 err.%)
Attention-56	**5.52**	5.89
Attention-92	**4.99**	5.35
Attention-128	**4.44**	5.57
Attention-164	**4.31**	7.18



Question

- Which one is false?

  a. With the same number of layers, attention networks have more parameters than CNNs.

  b. Attention networks can dynamically adjust the feature in each layer.

  c. CNN is a special case of attention network when the output of the mask branch is 0.

  d. The size of the receptive field of the mask branch is smaller than the trunk branch.

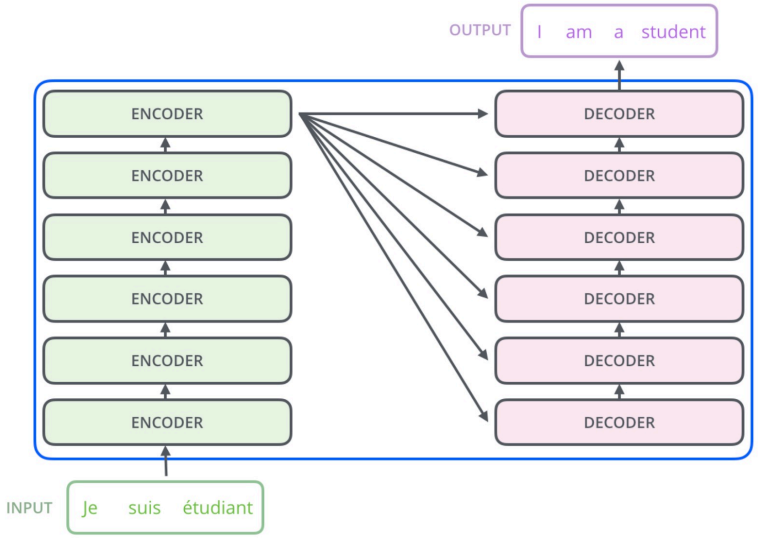
Demos

Will do demos in Jupyter notebooks

Part 4
Transformer

Transformer

- Transformers are originally proposed for Natural Language Processing (NLP)
- Intuitively, transformer tries to capture the interactions between different words.
- Transformer consists of multiple encoders and decoders.
- Compared with CNNs, transformers require much less computational resources.



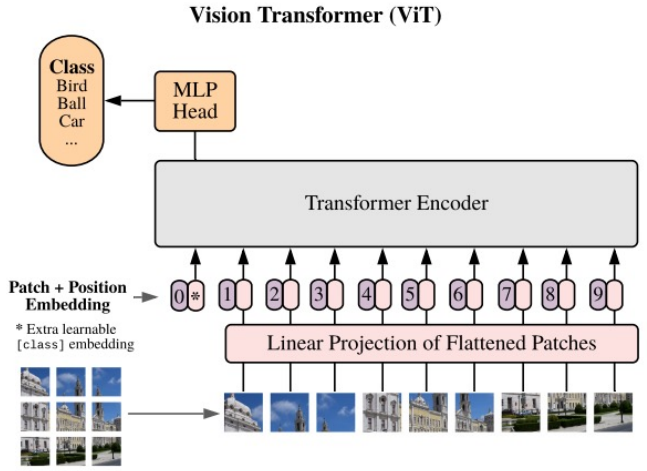
Transformers for Image Classification

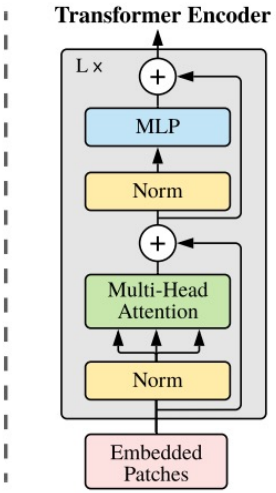
- Split an image into patches:
  - Image patches are treated the same way as tokens (words) in an NLP application.

- Flatten the patches

- Produce lower-dimensional linear embeddings from the flattened patches

- Add positional embeddings

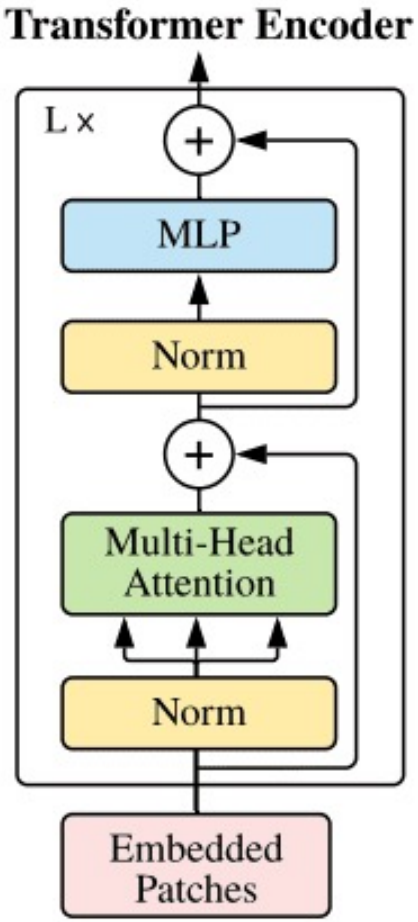




- Feed the sequence as an input to a standard transformer encoder

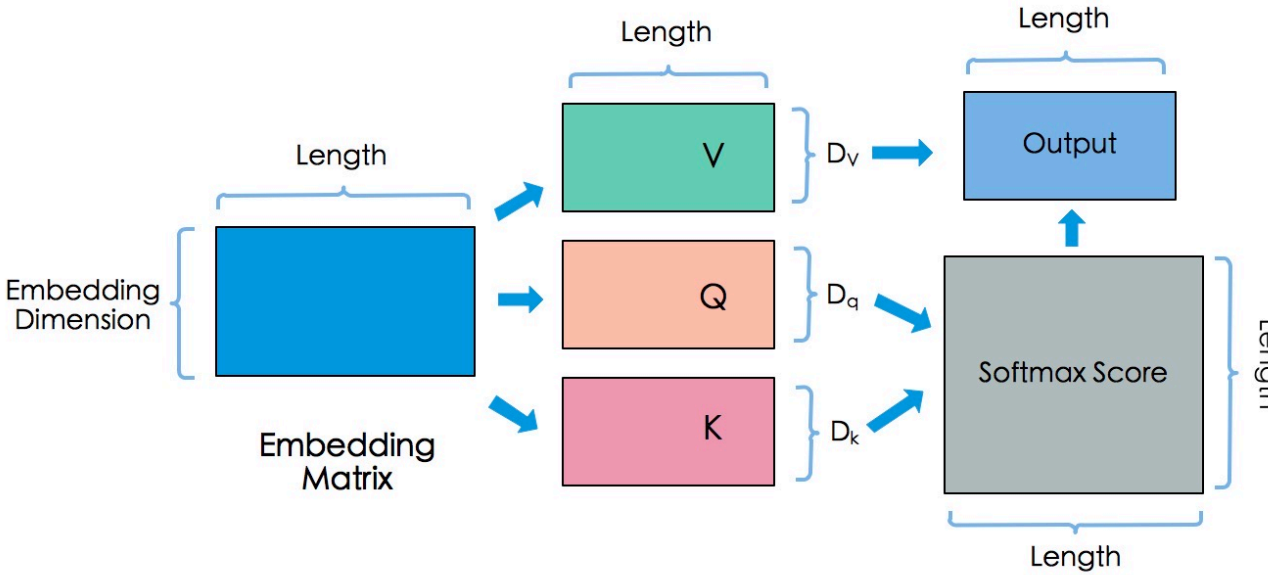
Transformer Encoder

- The embedded patches are normalized via layer-normalization
- The normalized embeddings are used as input for self-attention layer
- The output of the self-attention layer is again normalized via layer-normalization
- A multi-layer perceptron is applied at the end of the encoder



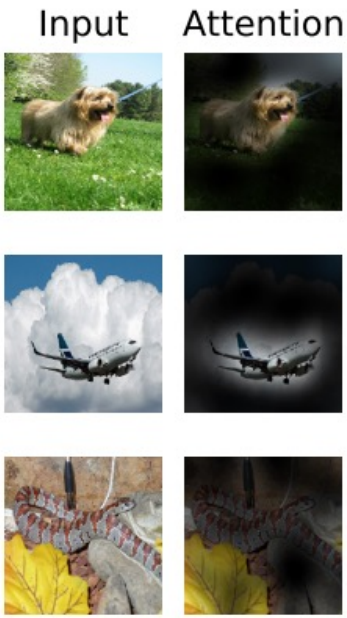
Self-Attention Layer

- The input embedding is multiplied by different matrices to produce query (Q) vectors, key (K) vectors and value (V) vectors.
- The output of the self-attention layer is computed as: $\text{Self-attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D_k}})\mathbf{V}$

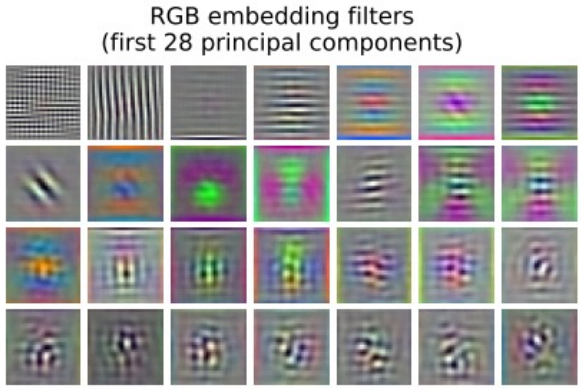


Inspecting Transformer

- The model attends to image regions that are semantically relevant for classification



Top principal components of the the learned embedding filters



Performance of Transformer

- Transformer models pre-trained on the JFT-300M dataset outperform ResNet-based baselines on all datasets, while taking substantially less computational resources to pre-train

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Demos

Will do demos in Jupyter notebooks

Summary

- Multi-layer percetrons (MLPs) are powerful universal function approximators.
- Convolutional neural networks (CNNs) are introduced to reduce the overfitting issue of MLPs.
- Attention networks incorporate attention mechanism in CNNs to locate the object in the input image.
- Transformers are built only with attentions which perform better than CNNs on large-scale datasets.

Exercise

Train a roof-type classifier based on CNNs and Transformers.