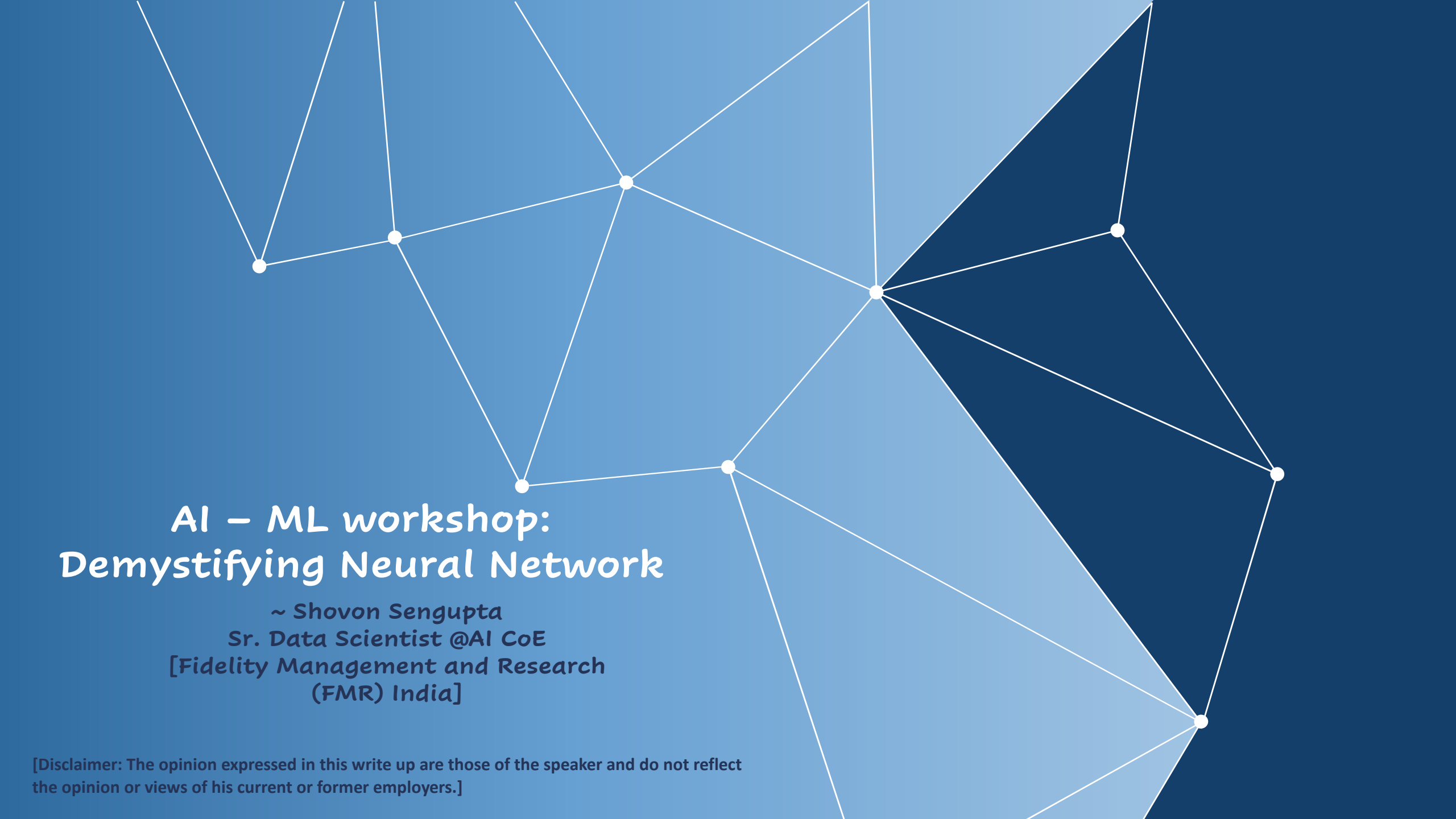
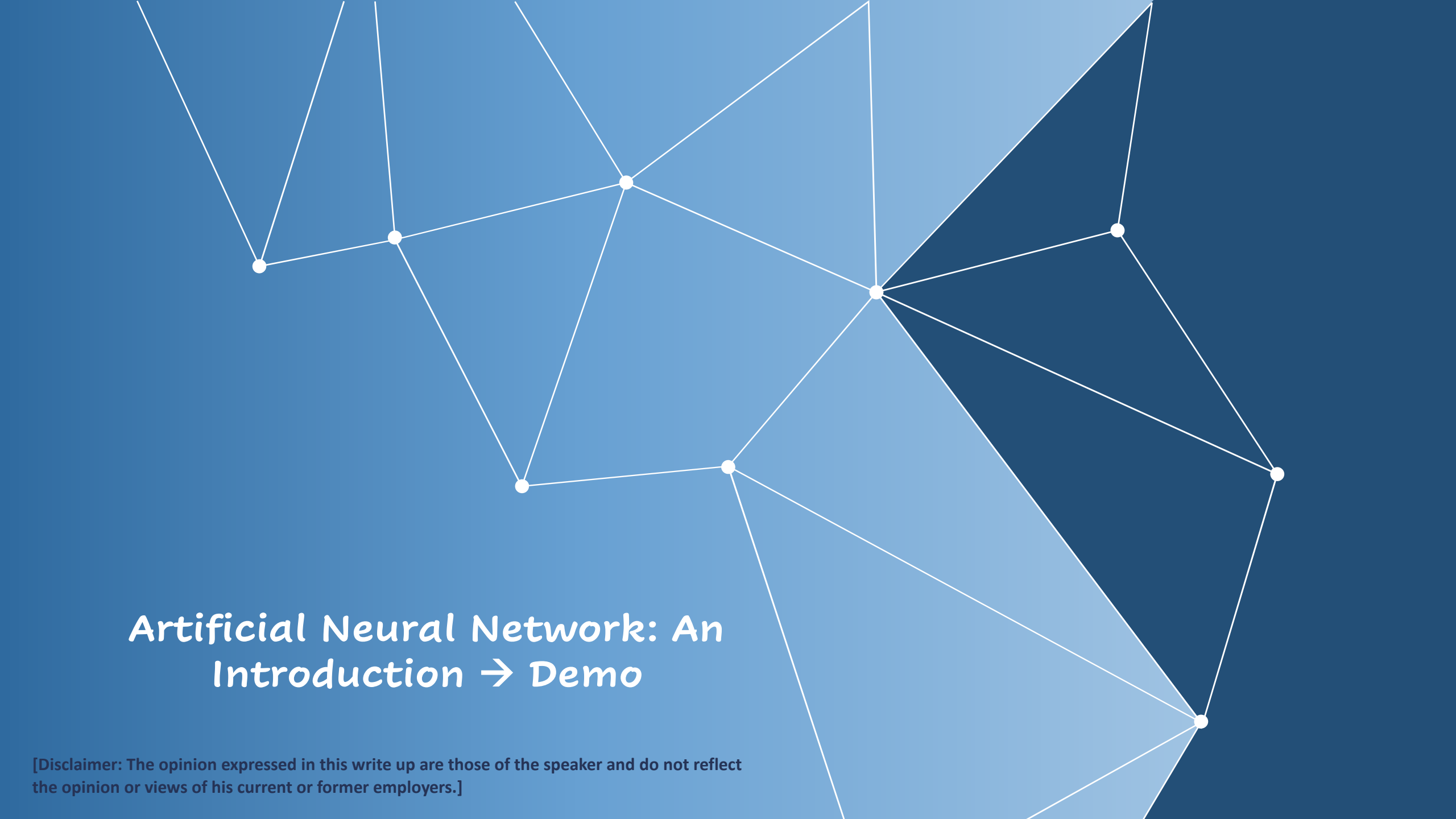


AI – ML workshop: Demystifying Neural Network

~ Shovon Sengupta
Sr. Data Scientist @AI CoE
[Fidelity Management and Research (FMR) India]

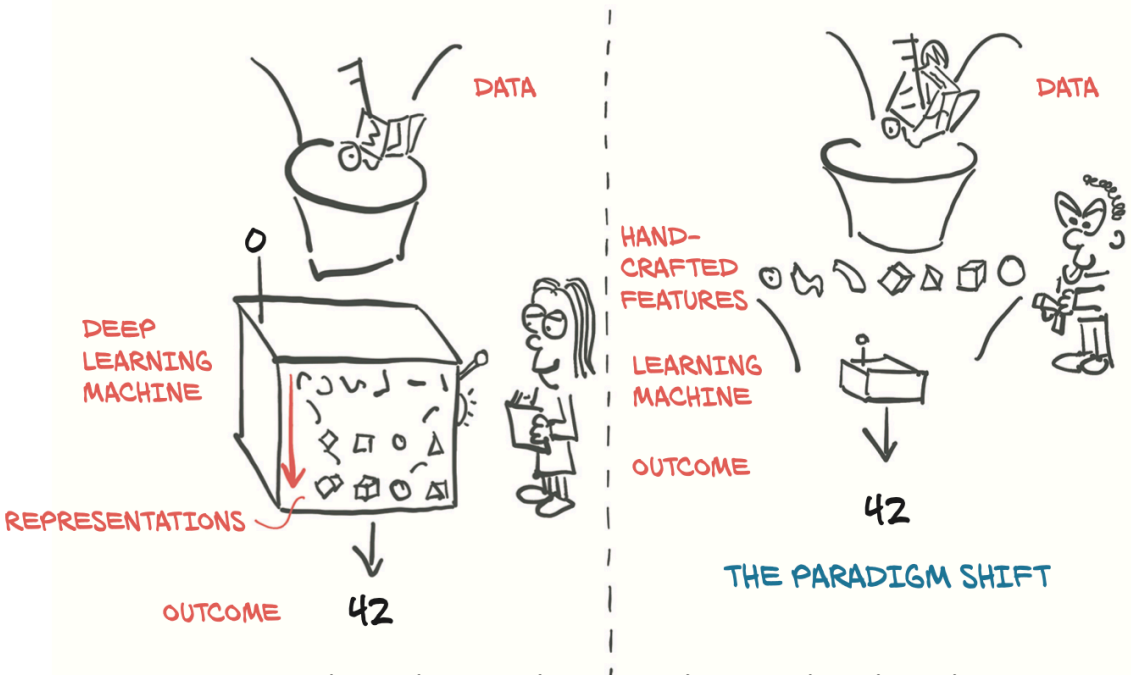


Artificial Neural Network: An Introduction → Demo

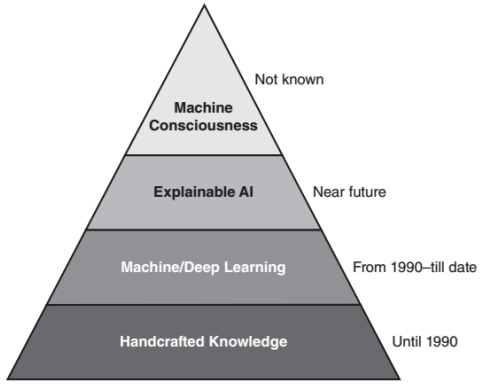
[Disclaimer: The opinion expressed in this write up are those of the speaker and do not reflect the opinion or views of his current or former employers.]

What is Deep Learning

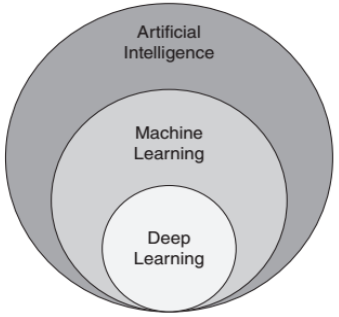
Deep learning is a type of machine learning in which large volumes of training data and artificial neural networks with multiple layers are used to achieve the machine learning tasks. We use the terminology 'Deep' because the number of processing layers is huge. Based on the many ways the layers are connected, we have many types of deep learning networks.



Deep learning exchanges the need to handcraft features for an increase in increase in data and computational requirements!

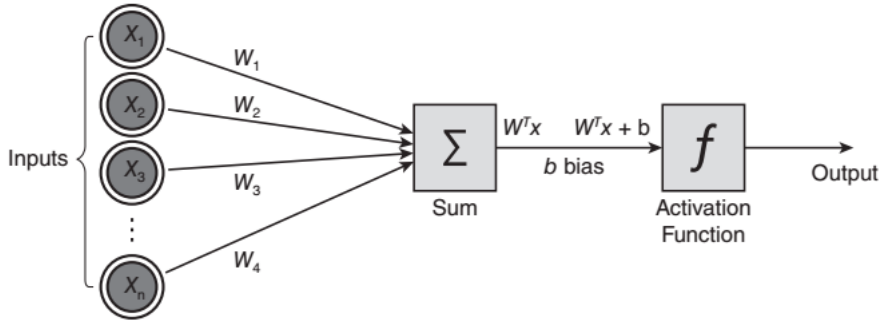


4 stages of AI

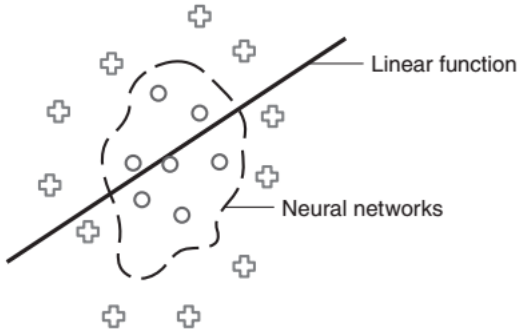


AI landscape

Overview: Deep Learning model → ANN



Basic building block of ANN



Activation Function and decision boundary

$$\text{mse} = \frac{1}{n} * \sum_{i=1}^{n} (y - \hat{y})^2$$

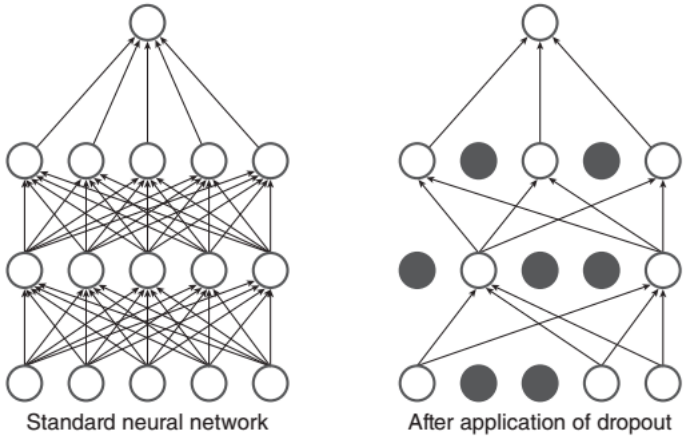
$$\text{binary cross entropy based on '1'} = -\frac{1}{n} * \sum_{i=1}^{n} y_i * \log(p_i)$$

$$\text{binary cross entropy} = -\frac{1}{n} * \sum_{i=1}^{n} y_i * \log(p_i) + (1 - y_i) * \log(1 - p_i)$$

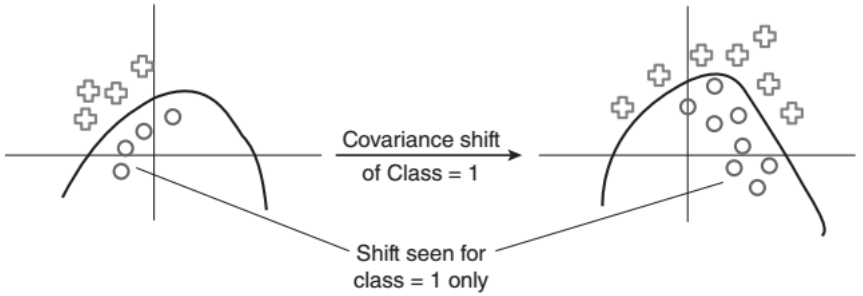
$$\text{categorical cross entropy} = -\frac{1}{n} * \sum_{i=1}^{n} \sum_{j=1}^{k} y_{ij} * \log(p_{ij})$$

i = number of observations, j = number of categories

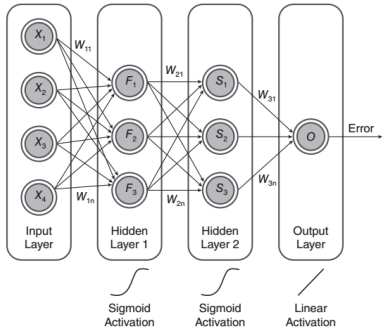
Loss function



Regularization: Drop out

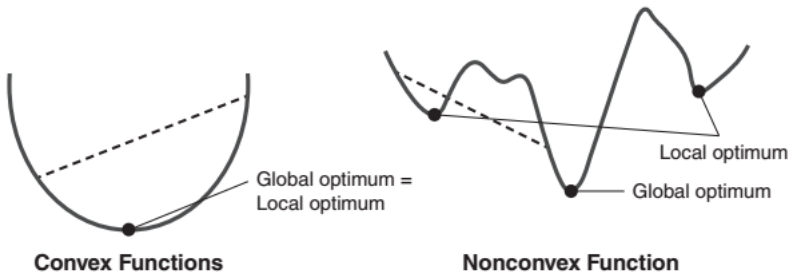


Regularization: Batch Normalisation



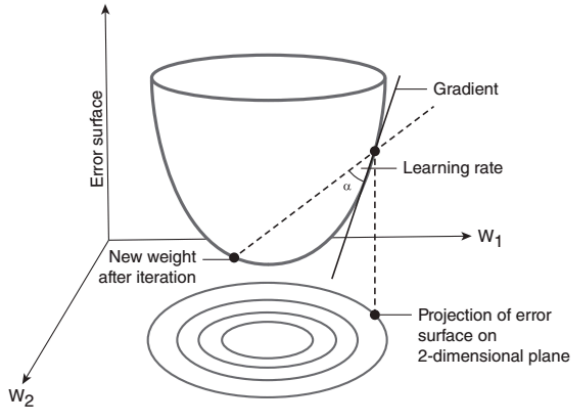
Forward propagation of NN

Convex function and optimizers

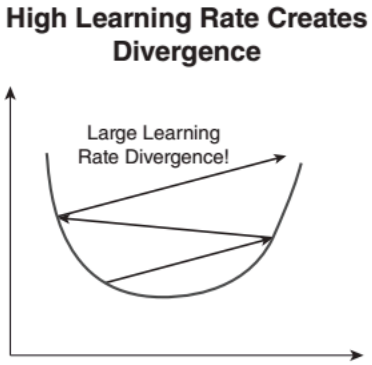


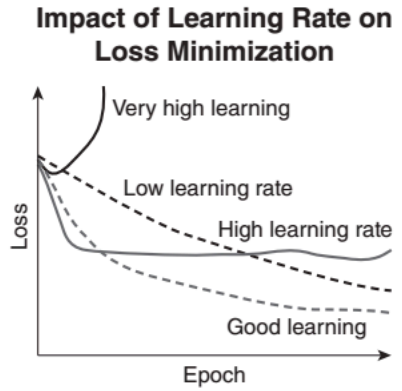
Example of Convex and Non- Convex function

$$W_{new} = W_{old} - \alpha * \nabla W_{old}$$

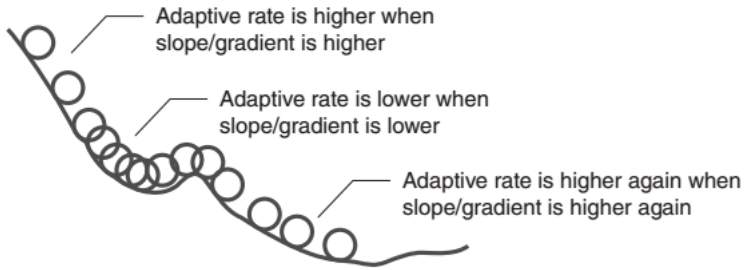


SGD – Stochastic Gradient Descent



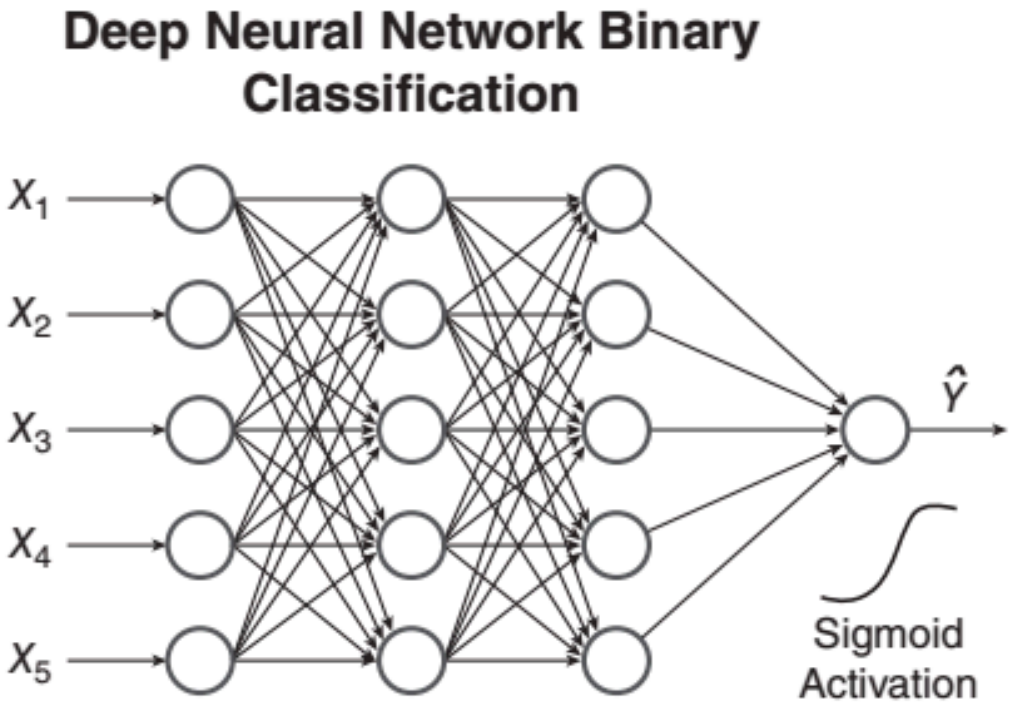


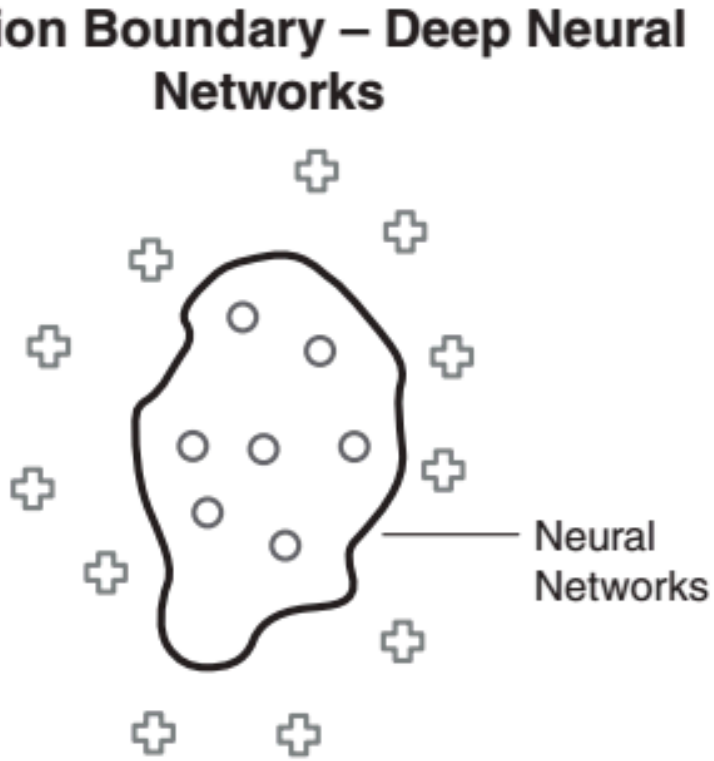
Impact of learning rate on convergence and loss minimization



Adoptive gradients and Learning rate

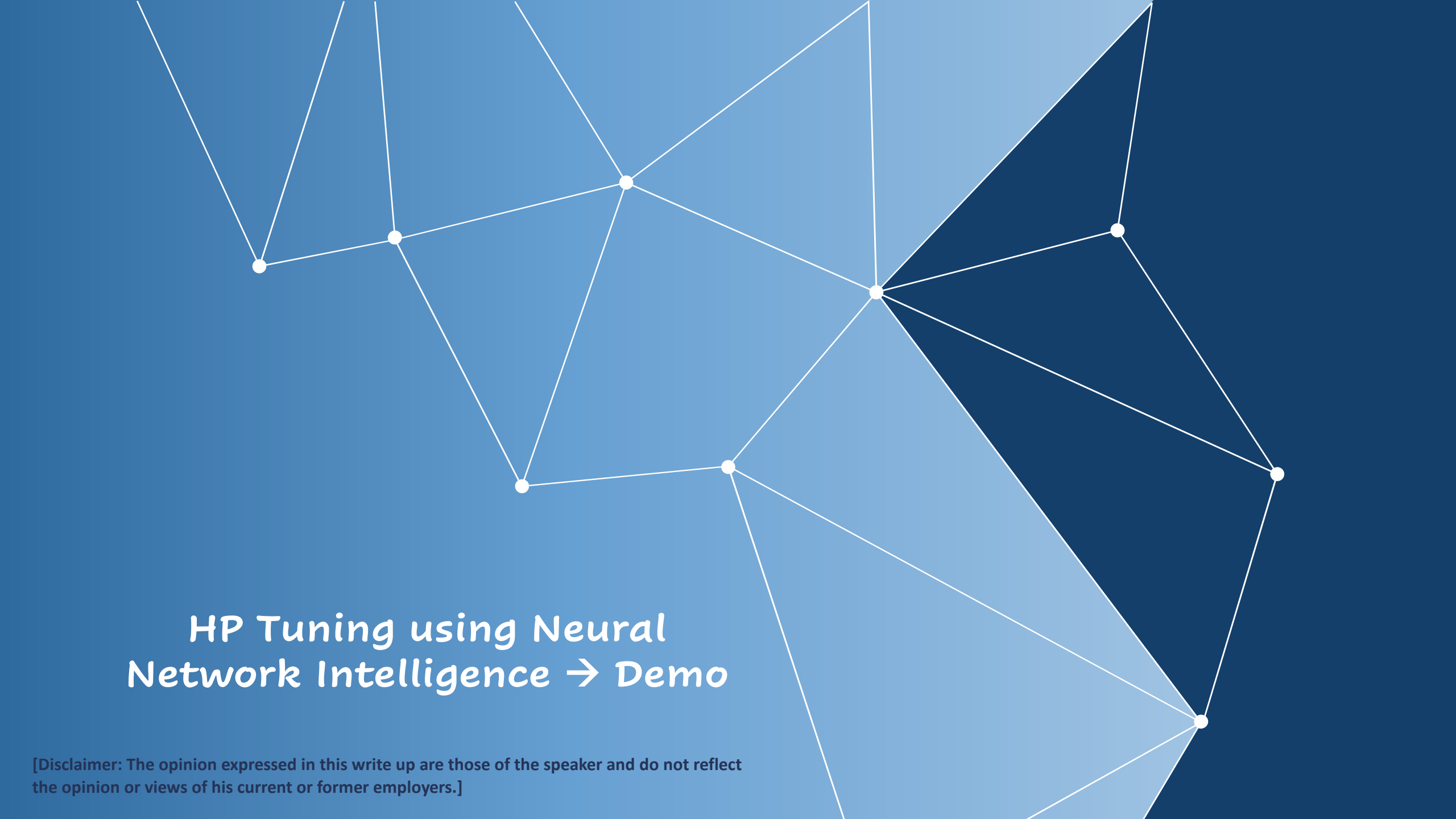
Binary classification and decision boundary





Case Study:

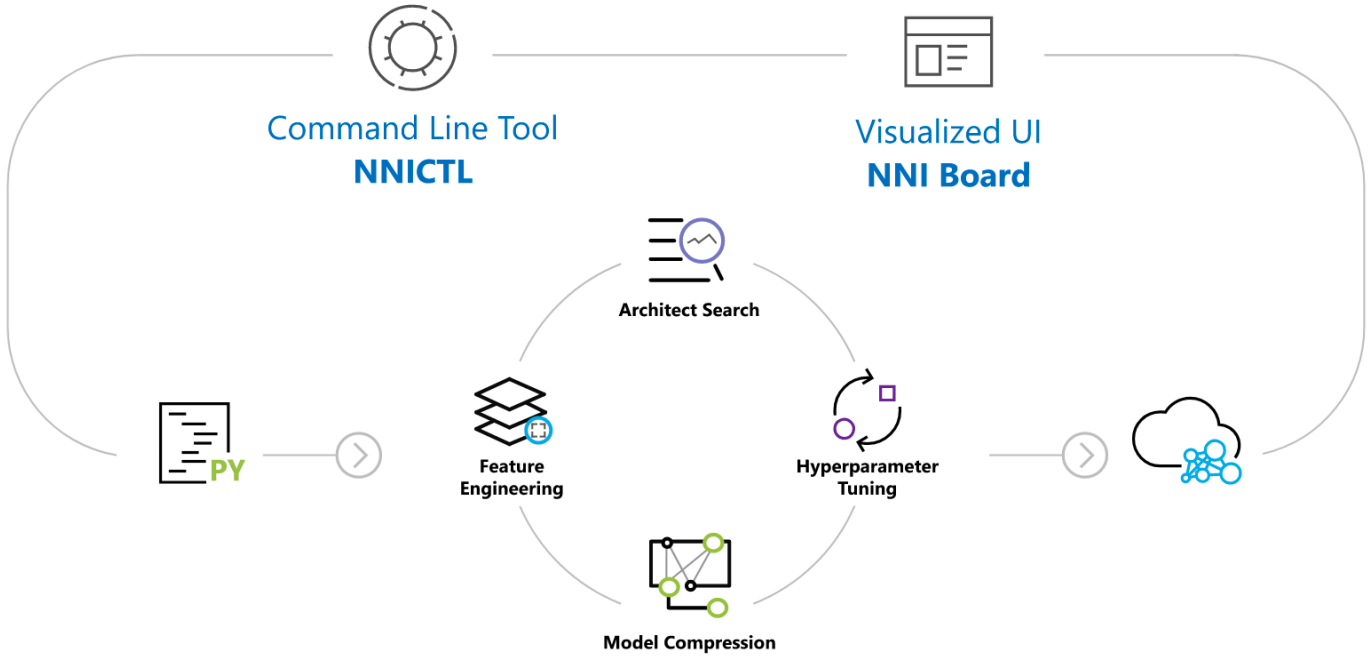
Application of ANN→ Demo



HP Tuning using Neural Network Intelligence → Demo

[Disclaimer: The opinion expressed in this write up are those of the speaker and do not reflect the opinion or views of his current or former employers.]

HP Tuning: "Neural Network Intelligence(NNI)"



NNI Capabilities by Microsoft

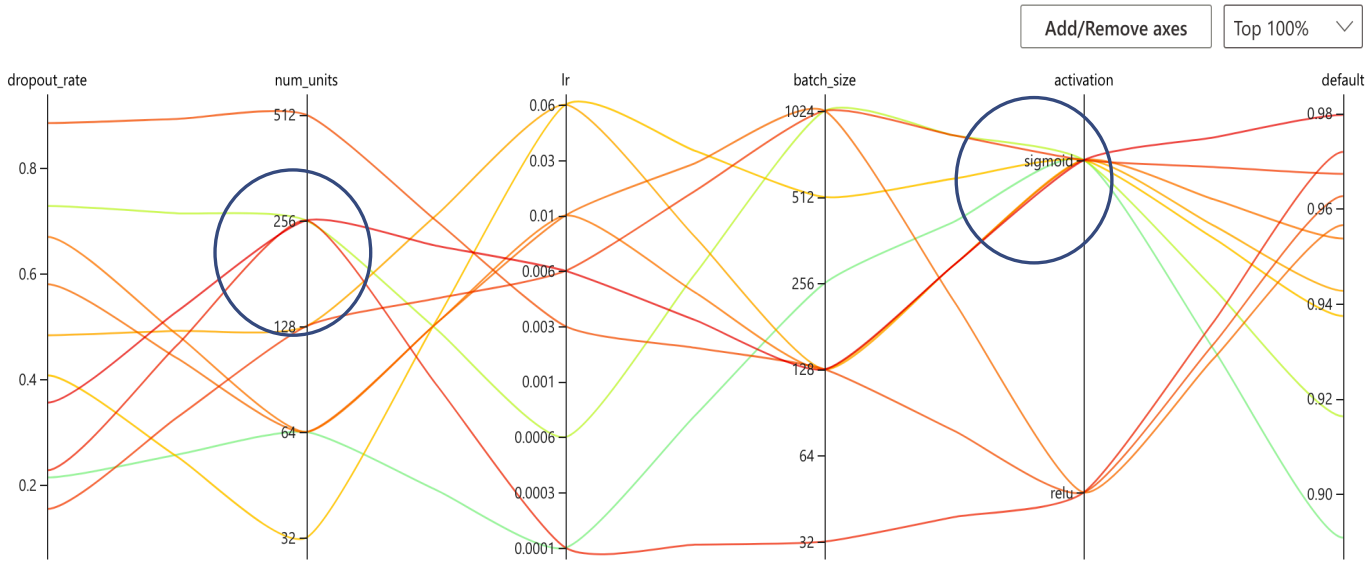
	Frameworks & Libraries	Algorithms	Training Services
	• **Supported Frameworks** ○ PyTorch ○ Keras ○ TensorFlow ○ MXNet ○ Caffe2 More... • **Supported Libraries** ○ Scikit-learn ○ XGBoost ○ LightGBM More...	Hyperparameter Tuning **Exhaustive search** ○ Random Search ○ Grid Search ○ Batch **Heuristic search** ○ Naïve Evolution ○ Anneal ○ Hyperband ○ PBT **Bayesian optimization** ○ BOHB ○ TPE ○ SMAC ○ Metis Tuner	• Local Machine • Remote Servers • AML(Azure Machine Learning) • **Kubernetes based services** ○ OpenPAI ○ Kubeflow ○ FrameworkController on K8S (AKS etc.) ○ DLWorkspace (aka. DLTS)

.....

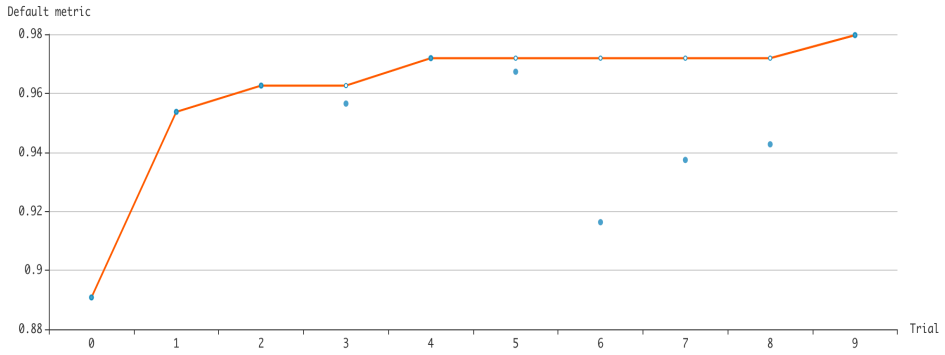
Available functionalities for HP Tuning

Source: Neural Network Intelligence

Optimal HPs: FFN model for MNIST data

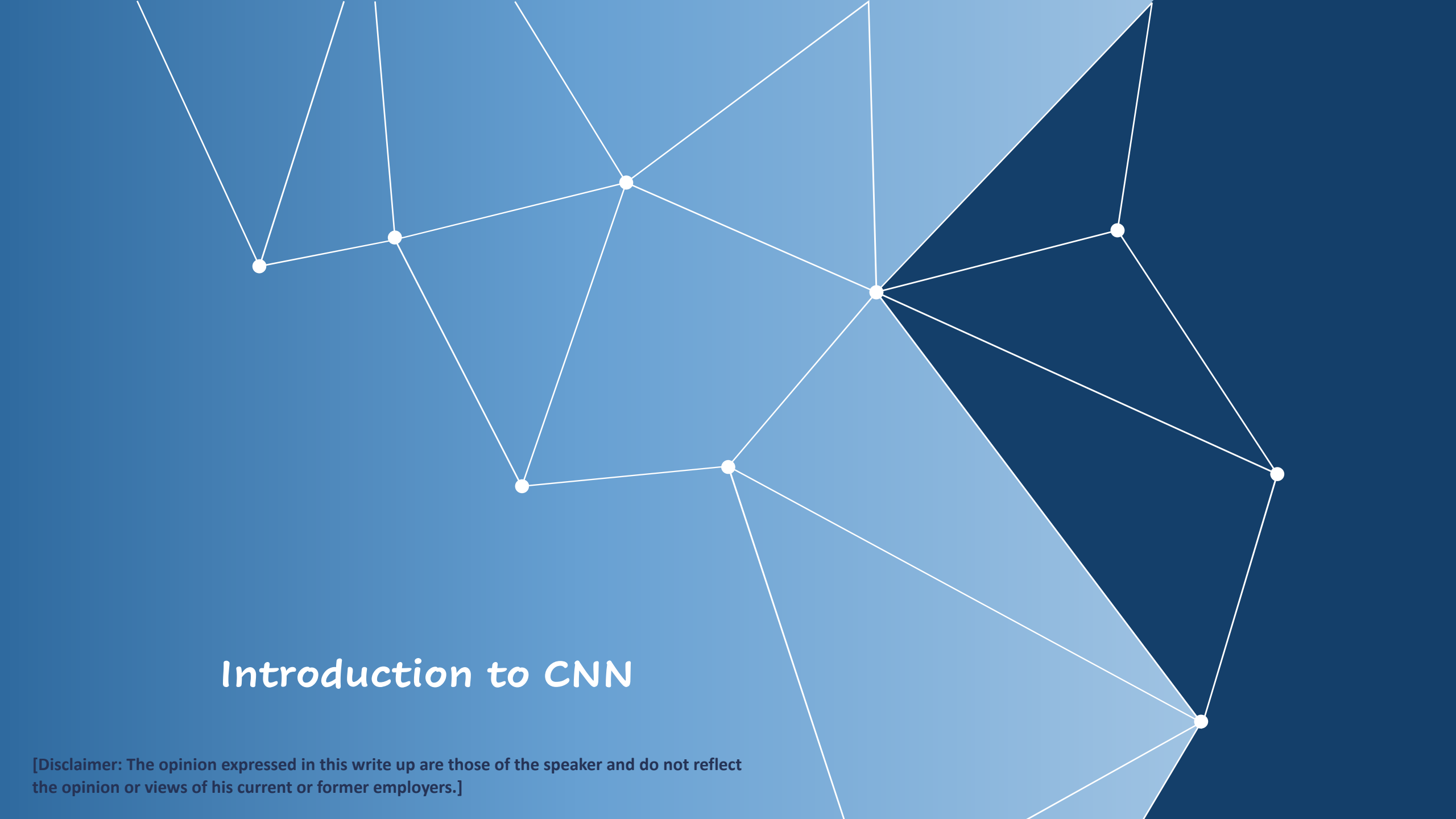


Trails Detail: HP Tuning for an FFN



Distribution of Default Metric: "Accuracy" across trials

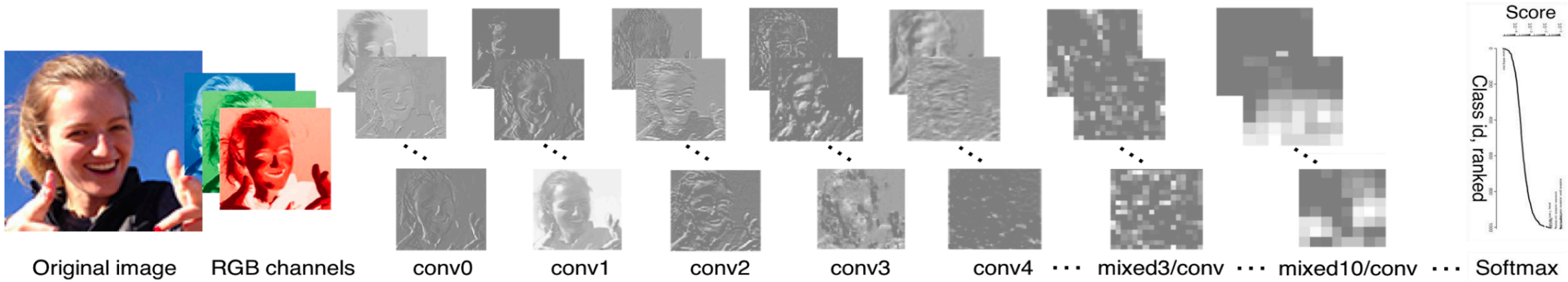
Source: Neural Network Intelligence



Introduction to CNN

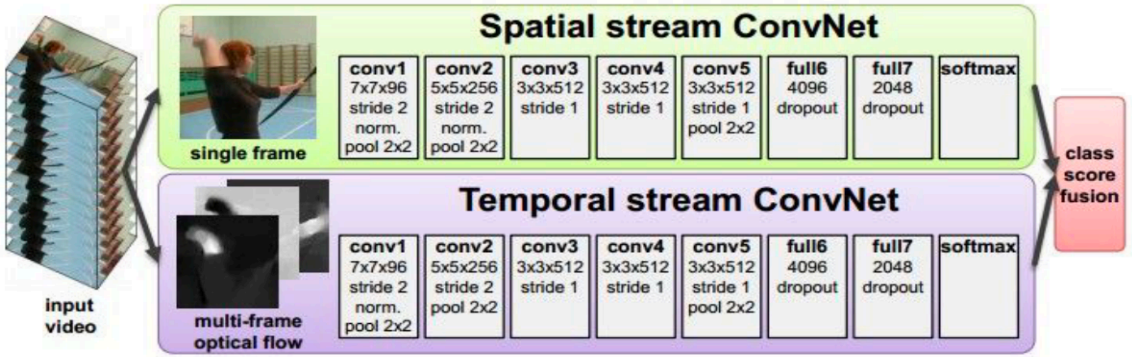
[Disclaimer: The opinion expressed in this write up are those of the speaker and do not reflect the opinion or views of his current or former employers.]

CNN are perhaps everywhere!



[Taigman et al. 2014]

Activations of inception-v3 architecture [Szegedy et al. 2015] to image of Emma McIntosh, used with permission. Figure and architecture not from Taigman et al. 2014.



[Simonyan et al. 2014]

Figures copyright Simonyan et al., 2014. Reproduced with permission.

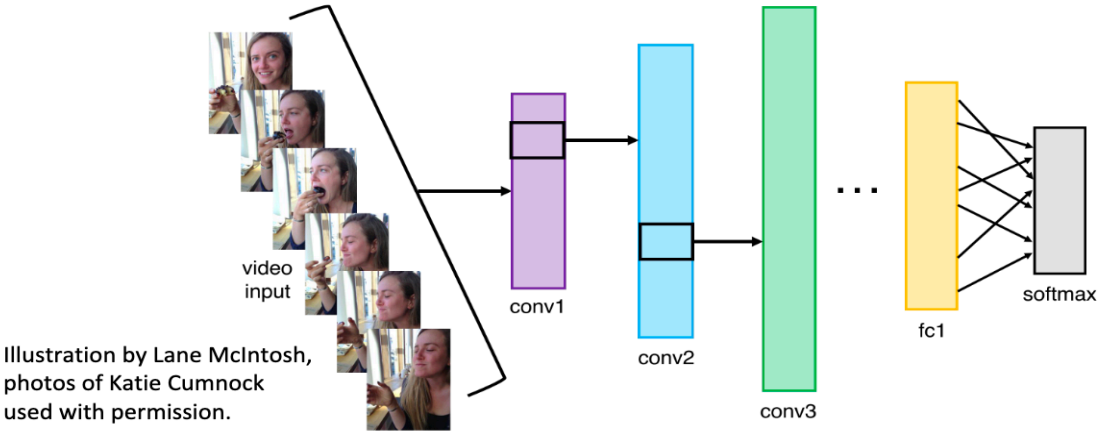
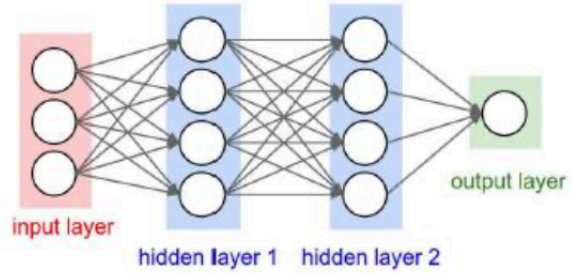


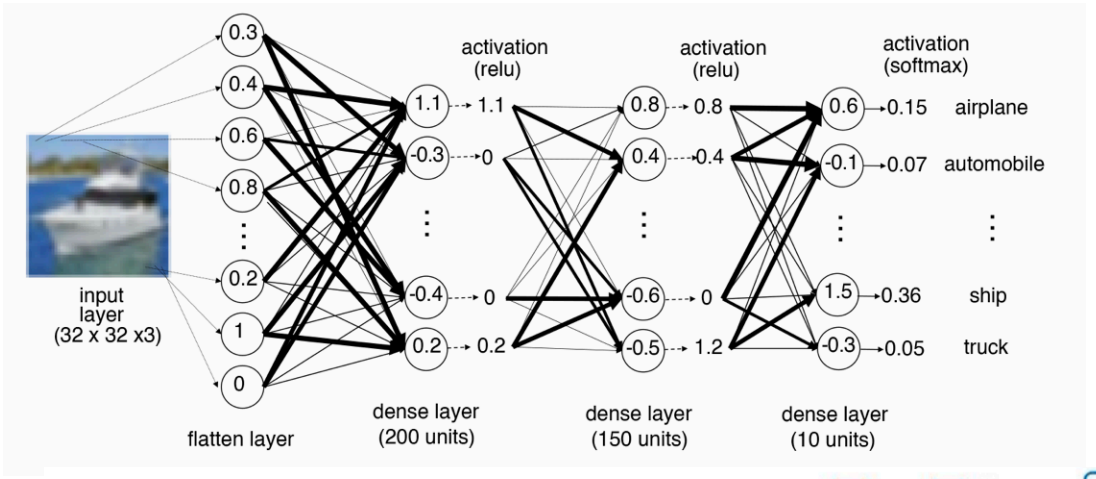
Illustration by Lane McIntosh, photos of Katie Cumnock used with permission.

Source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

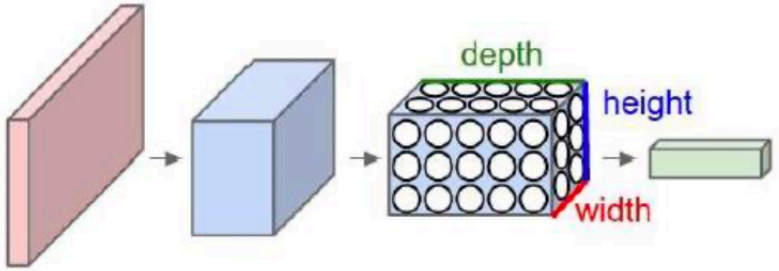
CNN – Convolutional Neural Network

Regular neural network (fully connected):

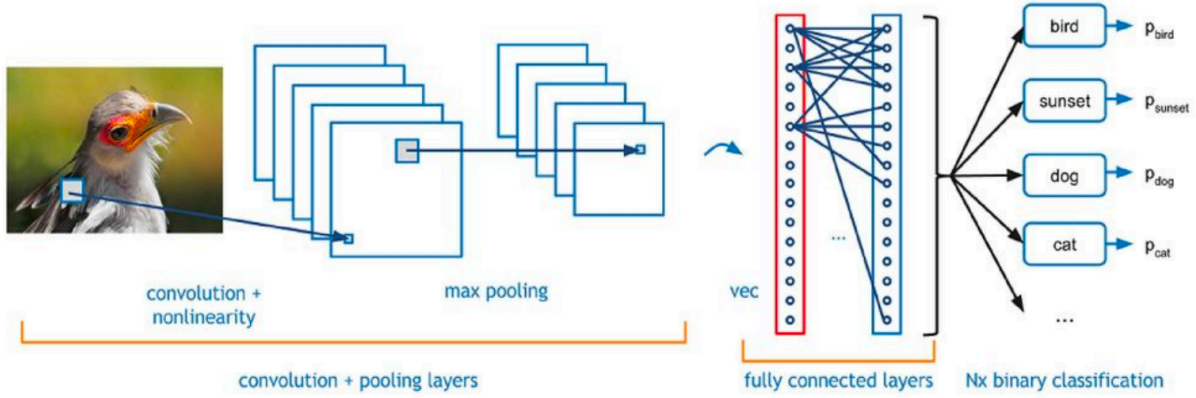




Convolutional neural network:

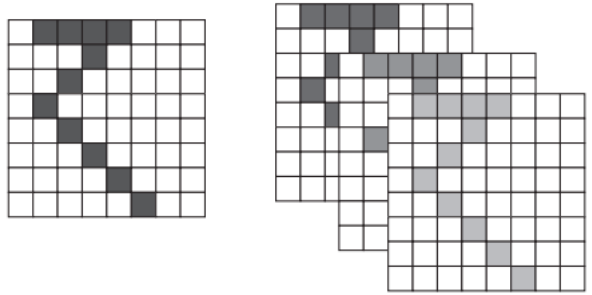


Each layer takes a 3d volume, produces 3d volume with some smooth function that may or may not have parameters.

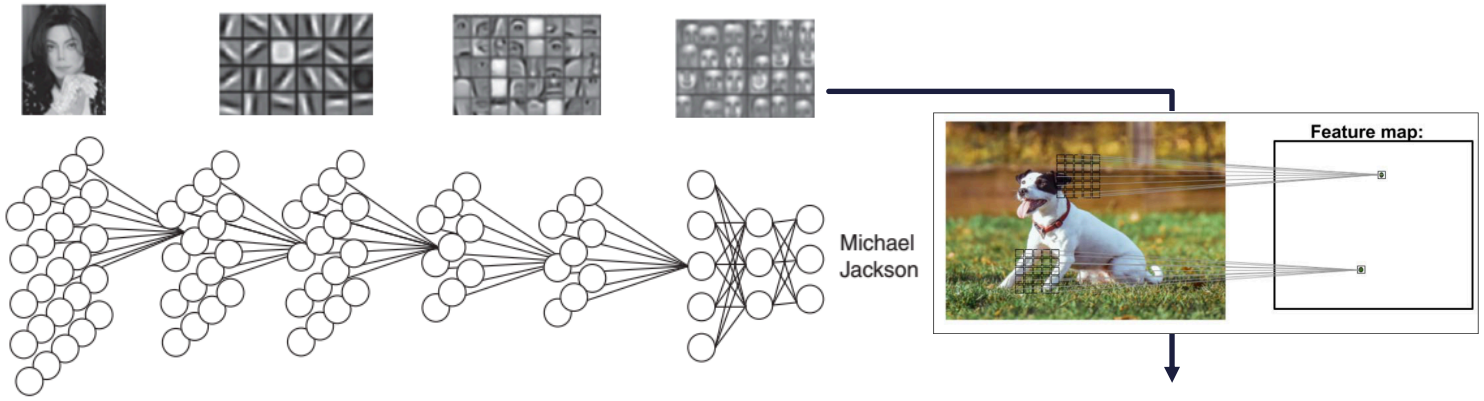


Source: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

CNN – Different Components



Example of Image Data



Formation of features over layers

255	0	124	207	145	111
20	215	132	97	88	41
122	45	255	41	60	54
2	26	32	3	225	212
33	246	263	110	96	75
48	63	184	34	64	33

Orignal grayscale image

1	1	0
0	0	1
1	0	1

Convolutional filter

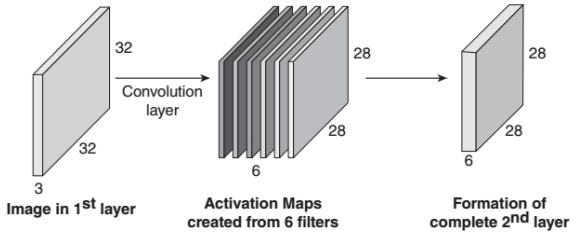
764			

764 = 255 × 1 + 0 × 1 + 124 × 0 + 20 × 0 + 215 × 0 + 132 × 1 + 122 × 1 + 45 × 0 + 255 × 1

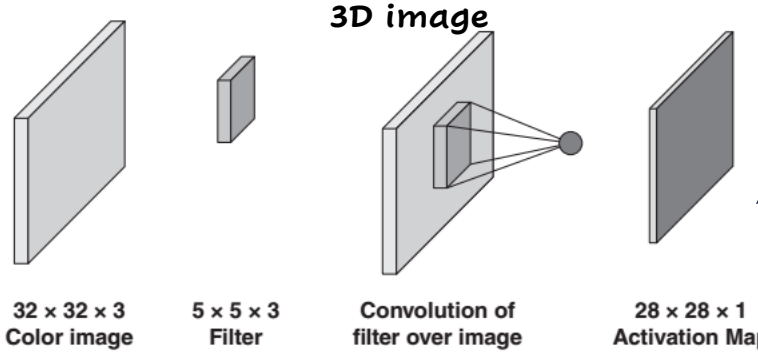
Feature vector

Convolutional Layers

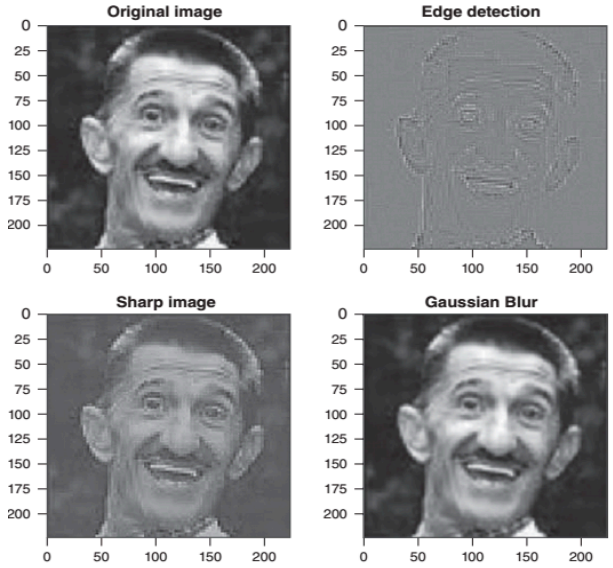
Convolutional layers created with 6 filters in 2nd layer



Application of Convolutional filter on greyscale 3D image



CNN – Pooling Layers and convolutional filters



Application of convolutional filters on Image

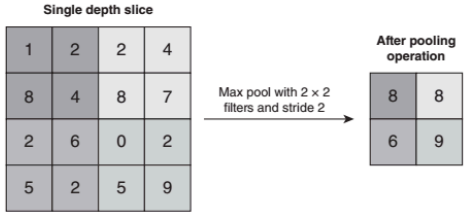
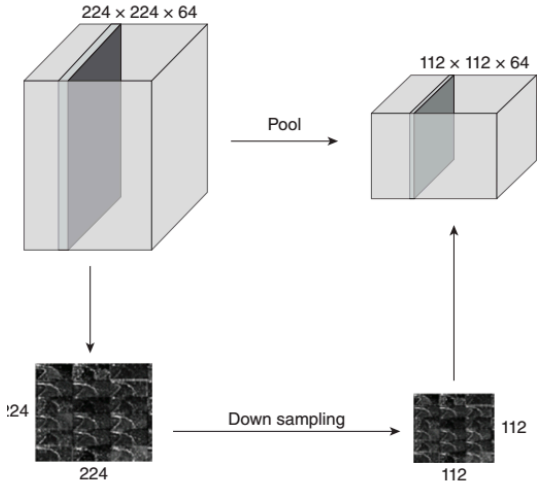
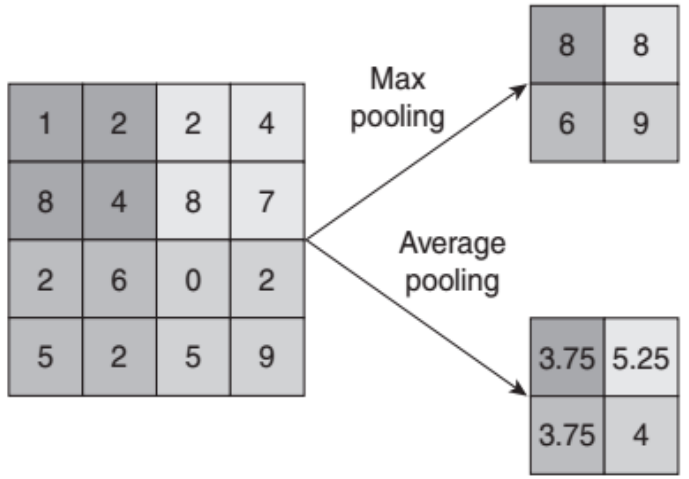


Figure 6.12 Maxpool Operation on Image Layer

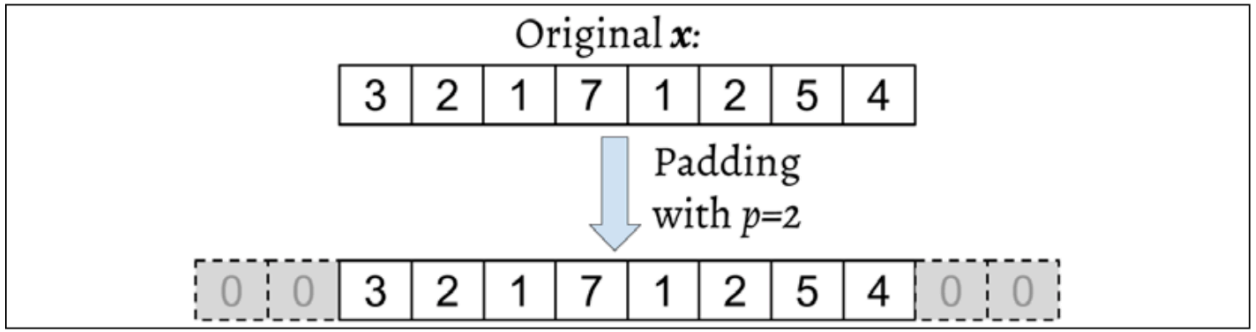


Max Pooling operations on 3D activation map



Max Pooling Vs. Average Pooling

CNN – Padding



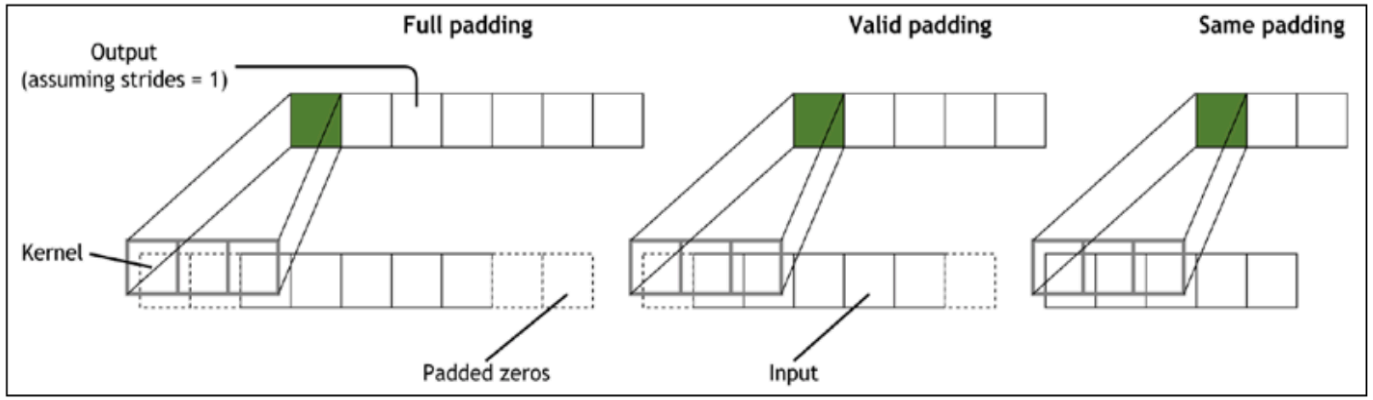
Basic Idea of padding

Determining the size of the convolution output

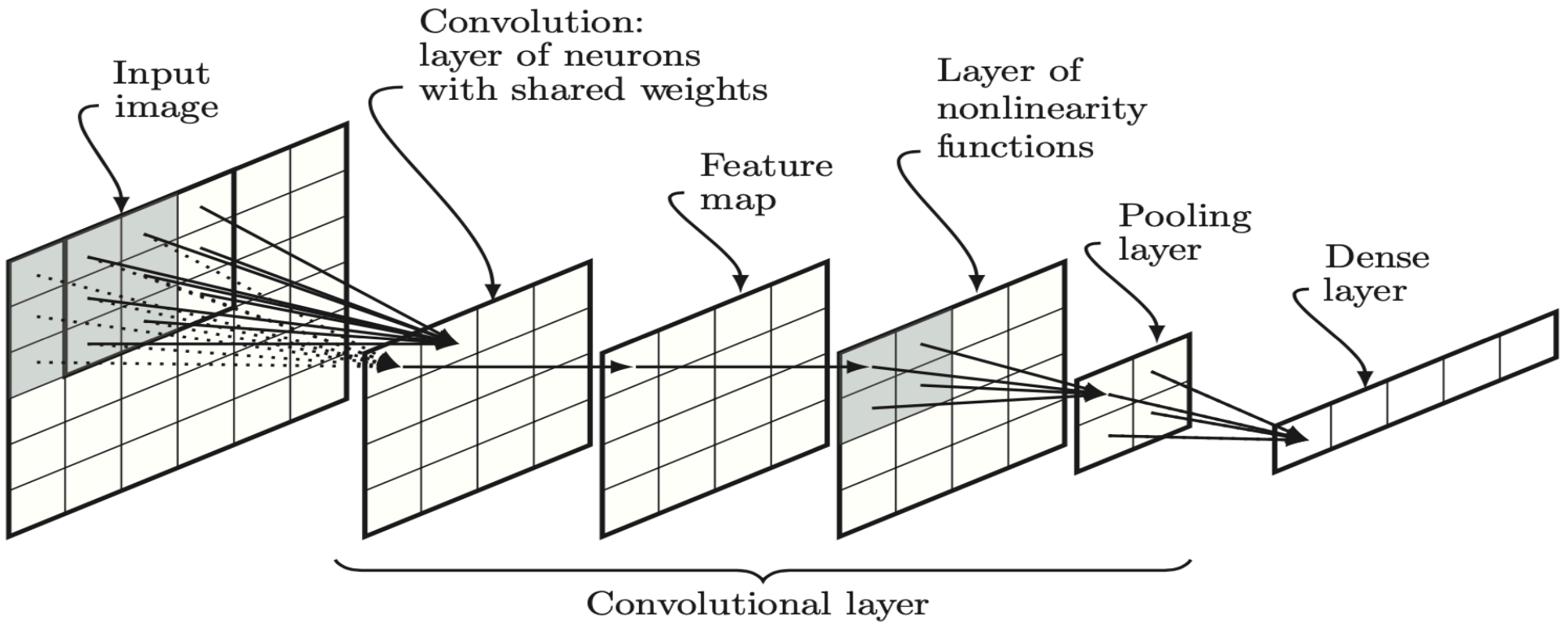
The output size of a convolution is determined by the total number of times that we shift the filter, $\boldsymbol{w}$, along the input vector. Let's assume that the input vector is of size n and the filter is of size m. Then, the size of the output resulting from $\boldsymbol{y} = \boldsymbol{x} * \boldsymbol{w}$, with padding, p, and stride, s, would be determined as follows:

$$o = \left\lfloor \frac{n + 2p - m}{s} \right\rfloor + 1$$

Example of 3 different padding modes for a simple 5 X 5 pixels input with a kernel size of 3 X 3 and stride 1

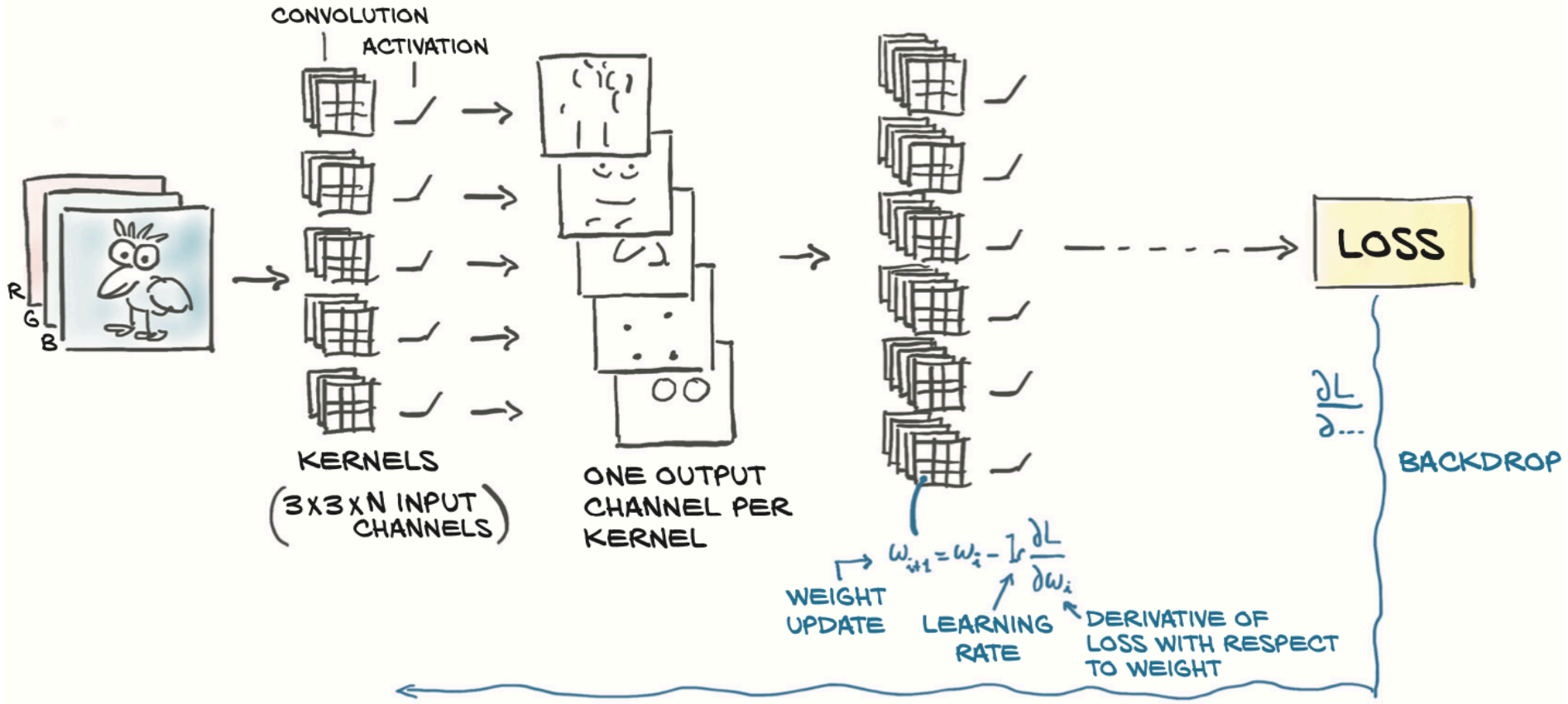


Basic architecture of CNN



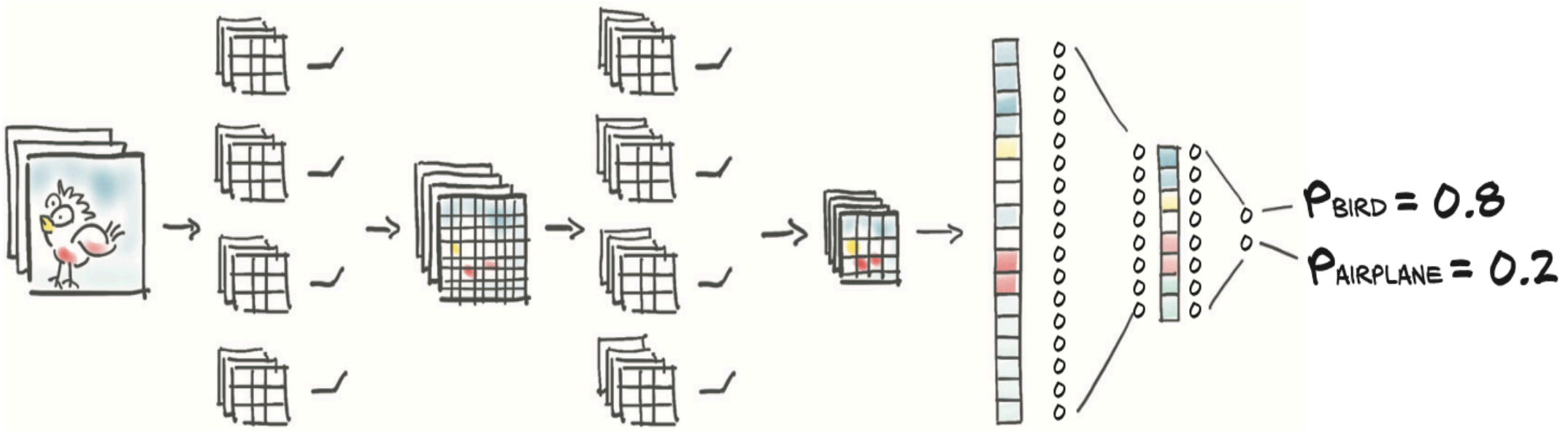
Source: Deep Learning By John D. Kelleher

Learning with Convolution



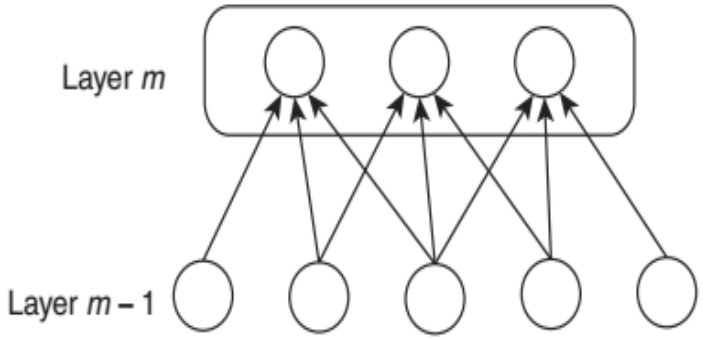
The process of learning with convolutions by estimating the gradient at the kernel weights and updating them individually in order to optimize for the loss

Shape of a CNN!

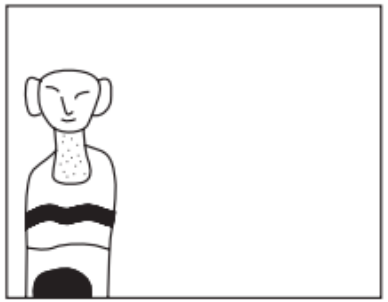


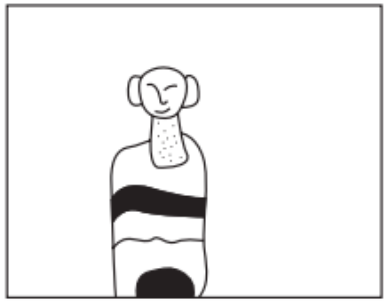
Shape of a typical CNN → An image if fed to a series of convolutions and max pooling modules and then straightened into a 1D vector and fed into a fully connected modules

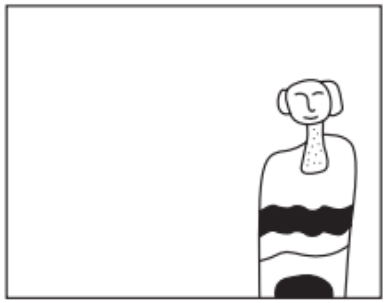
Unique properties of CNN!



Weight Sharing







Translation Invariance

Examples of a few different architectures of CNN

- LeNet
- AlexNet
- ZFNet
- GoogleNet
- VGGNet
- ResNet
- DenseNet

More on CNN Architecture

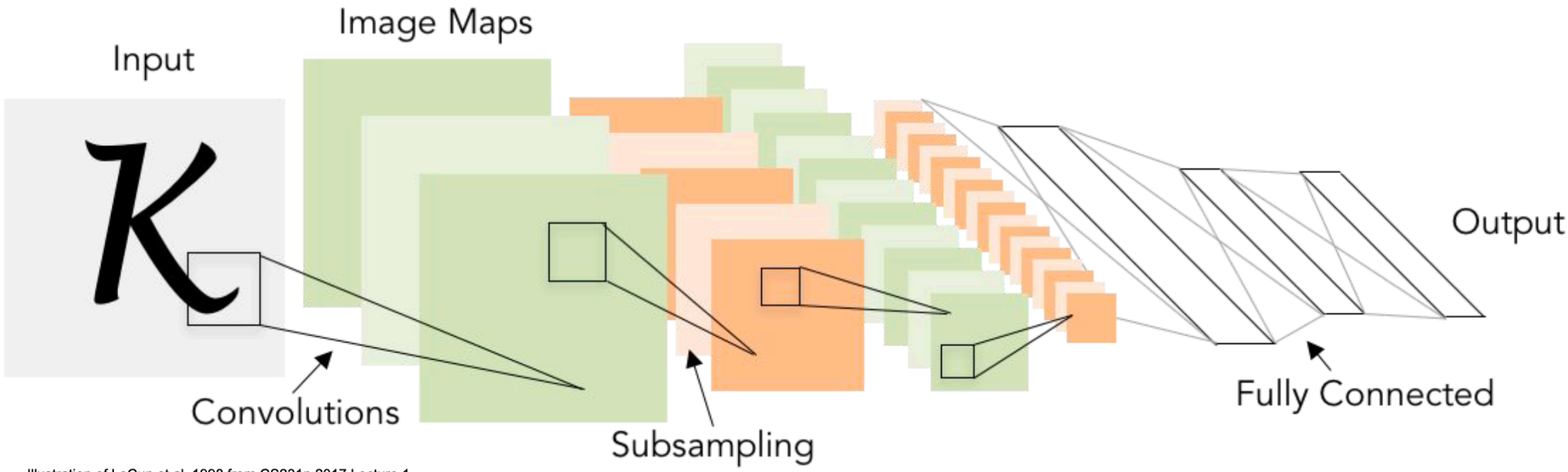
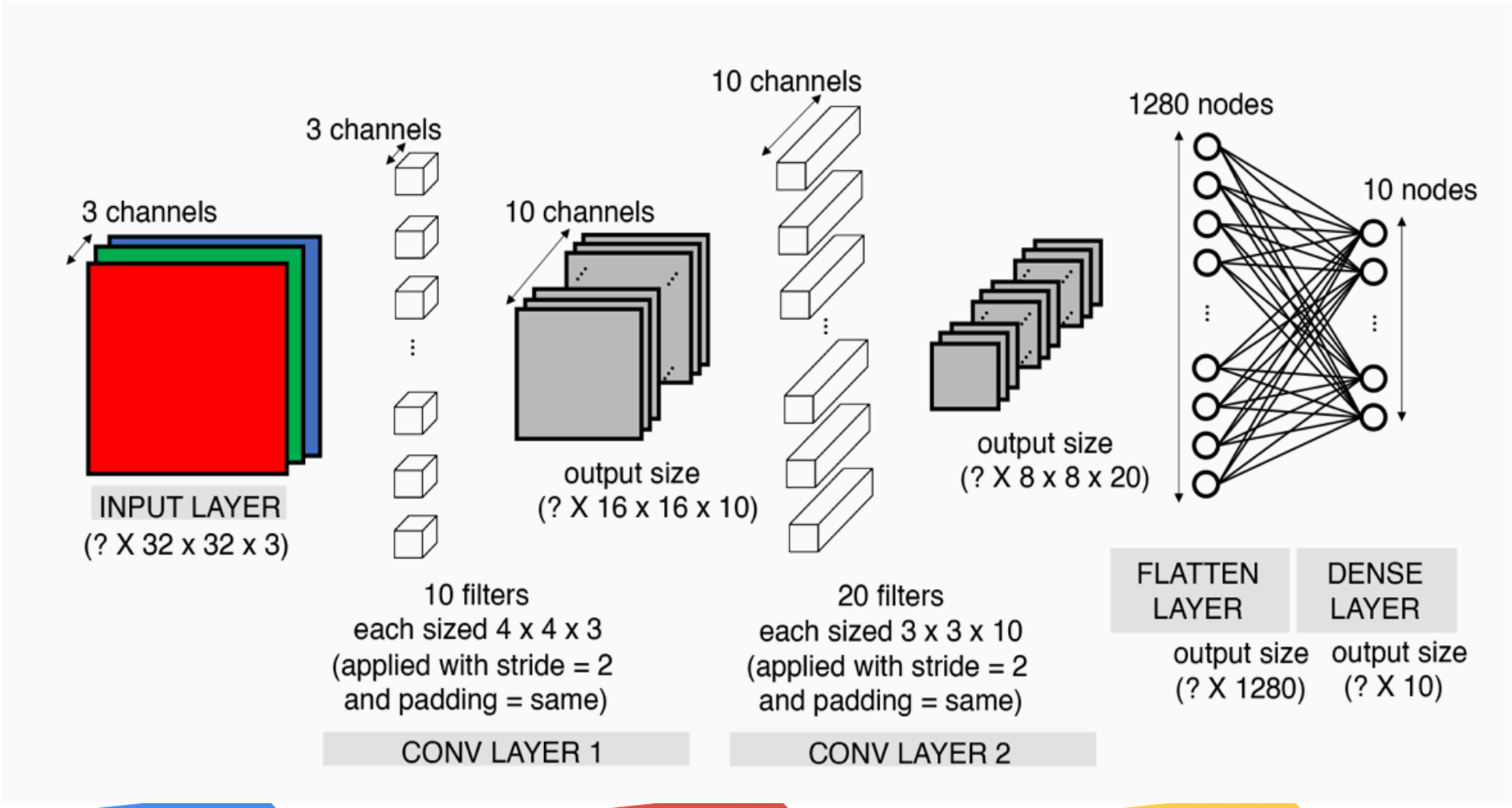


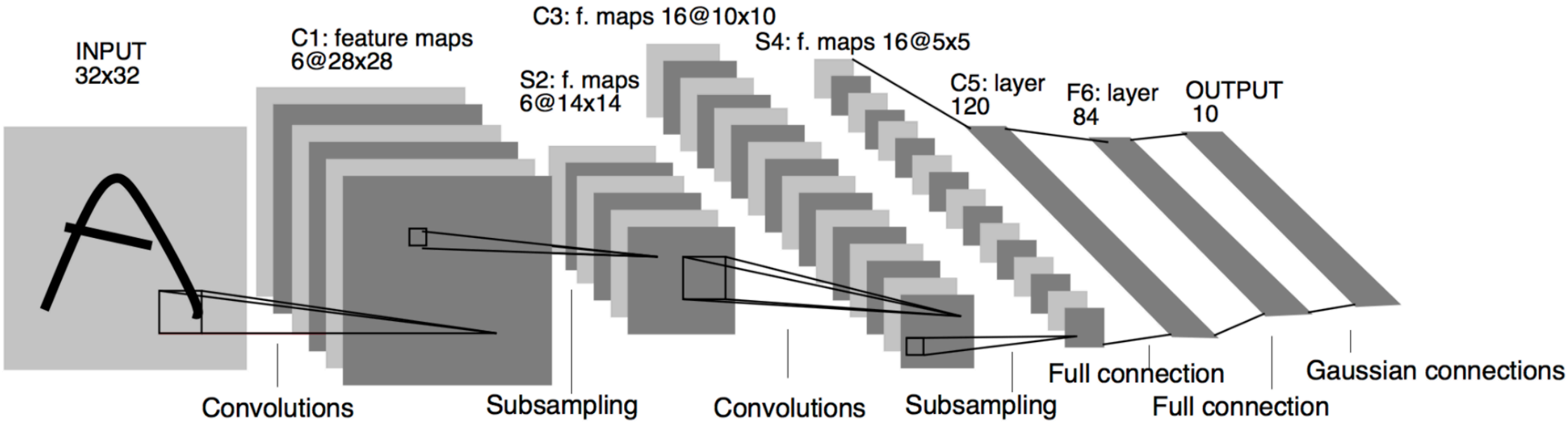
Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Source: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

CNN contd.


3 channels
INPUT LAYER
(? X 32 x 32 x 3)
3 channels
10 filters
each sized 4 x 4 x 3
(applied with stride = 2
and padding = same)
CONV LAYER 1
10 channels
output size
(? X 16 x 16 x 10)
10 channels
20 filters
each sized 3 x 3 x 10
(applied with stride = 2
and padding = same)
CONV LAYER 2
output size
(? X 8 x 8 x 20)
1280 nodes
10 nodes
FLATTEN LAYER
output size
(? X 1280)
DENSE LAYER
output size
(? X 10)

Example: LeNet-5



Introduced in 1998 by LeCun et al., LeNet was the first deep convolutional architecture. It was used to perform OCR by several banks to recognize handwritten numbers on cheques digitized in **32 × 32 grayscale images**. Though its ability to solve was admirable, it was constrained by the availability of computing resources at that time because of its high computational costs.

Source: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

Case Study

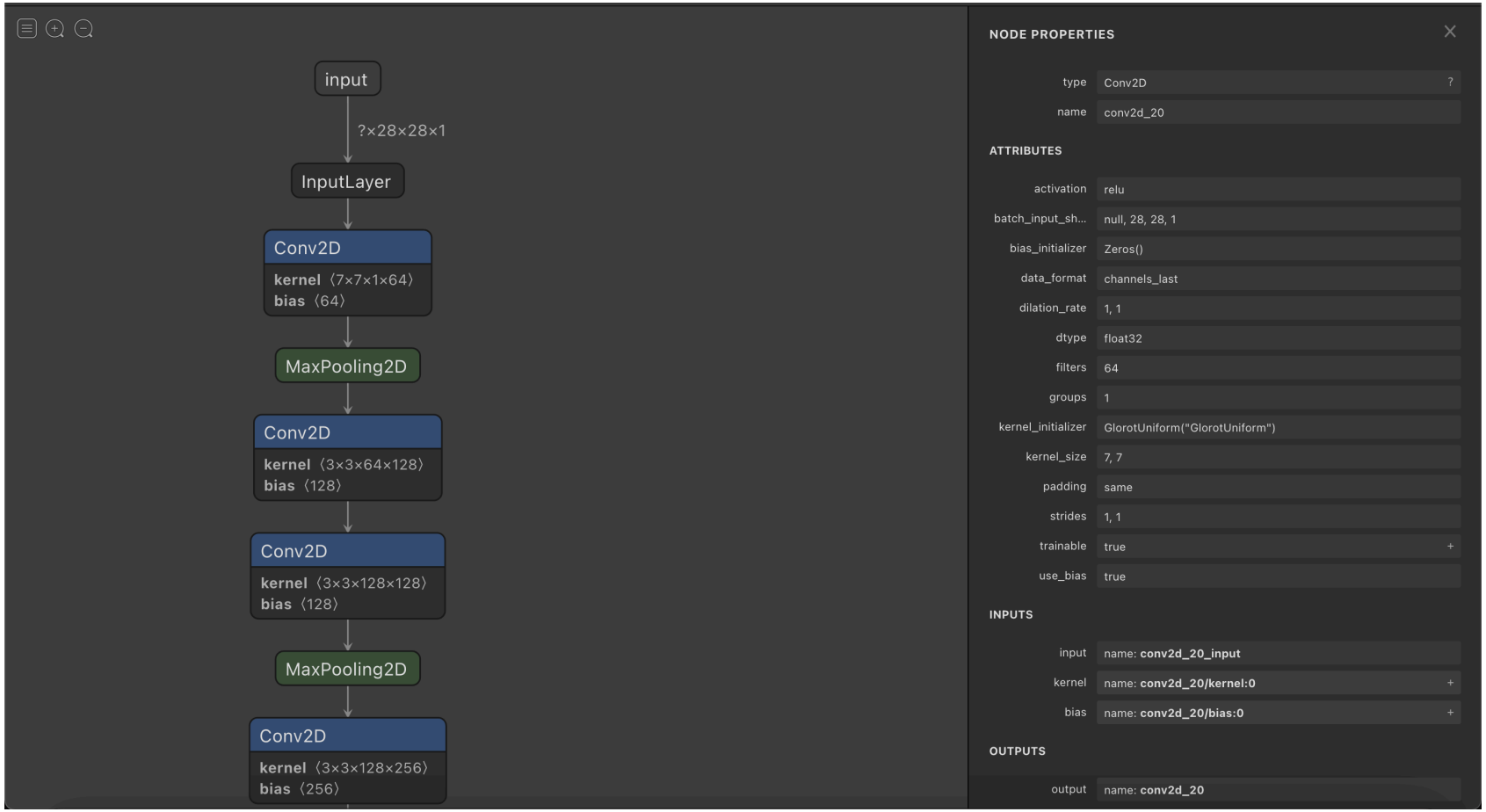
Application of CNN and Advanced CNN→ Demo

Visualization: CNN Model

Visualization through "NETRON"→ Demo

[Disclaimer: The opinion expressed in this write up are those of the speaker and do not reflect the opinion or views of his current or former employers.]

Visualization of a CNN Model through NETRON

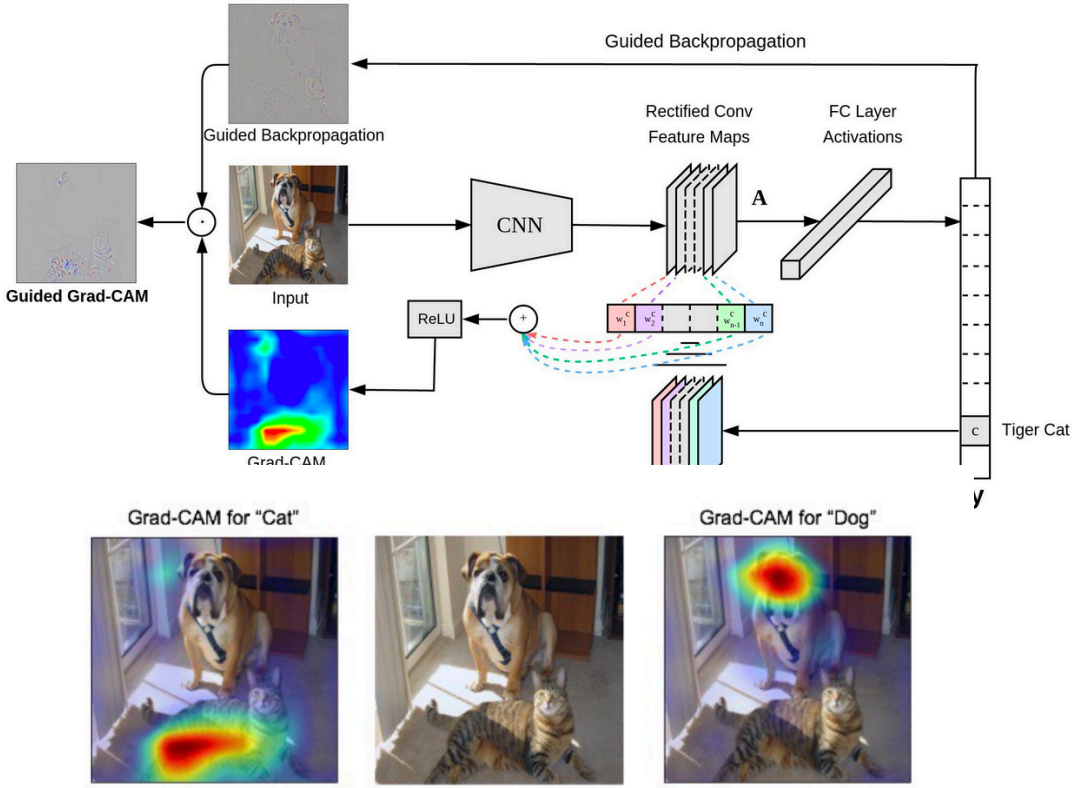


Explainable AI

Visual Interpretation: CNN Model → Demo

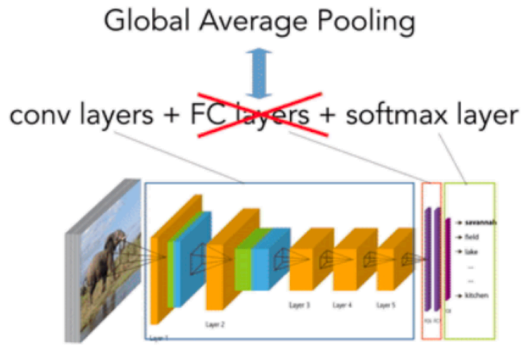
[Disclaimer: The opinion expressed in this write up are those of the speaker and do not reflect the opinion or views of his current or former employers.]

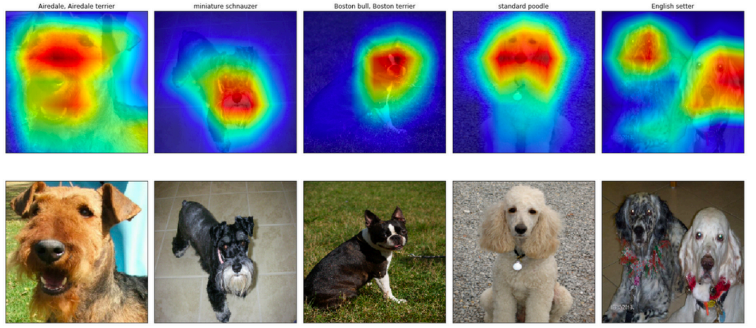
Visual Explanation: CNN through Grad-CAM



results in a coarse heat-map of the same size as the convolutional feature maps (14×1414×14 in the case of last convolutional layers of VGG and AlexNet networks)

CAM – Class activation Mapping



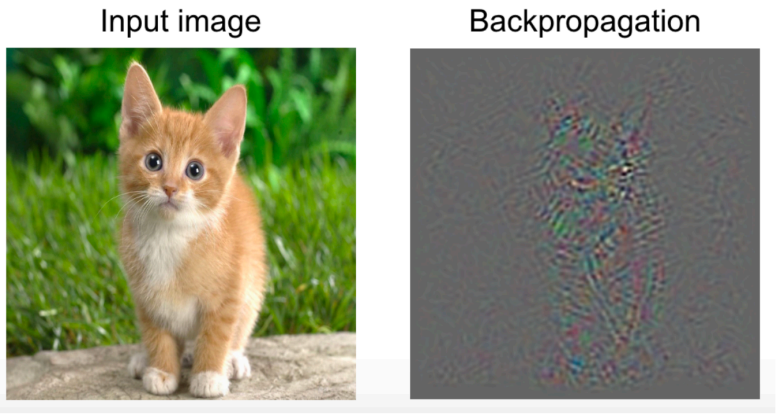


https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/

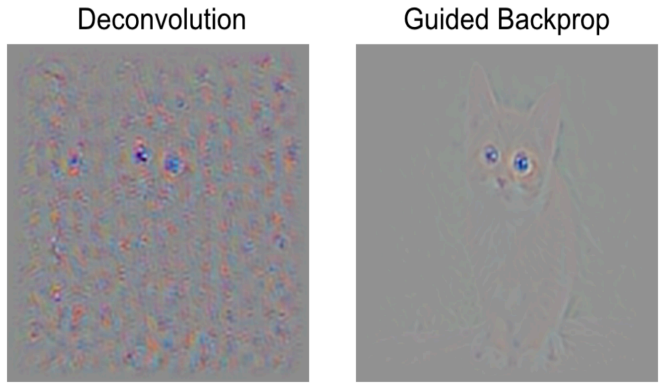
Example : Gradient Weighted Class Activation Mapping

Source: Grad-CAM: Visual Explanations from Deep Networksvia Gradient-based Localizatio

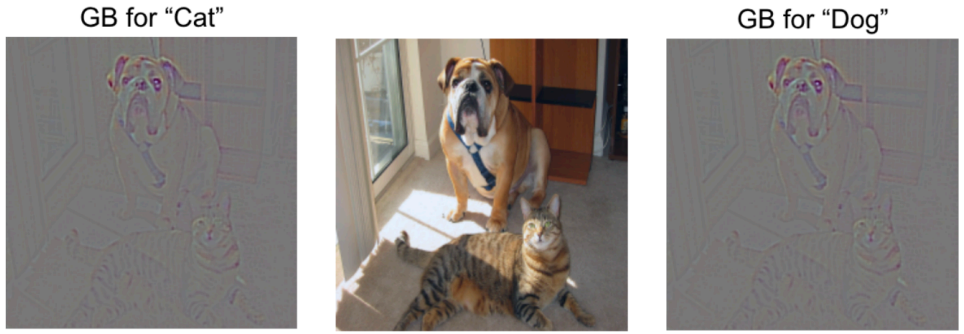
Overview of CAM/Grad-CAM



Back-propagation step

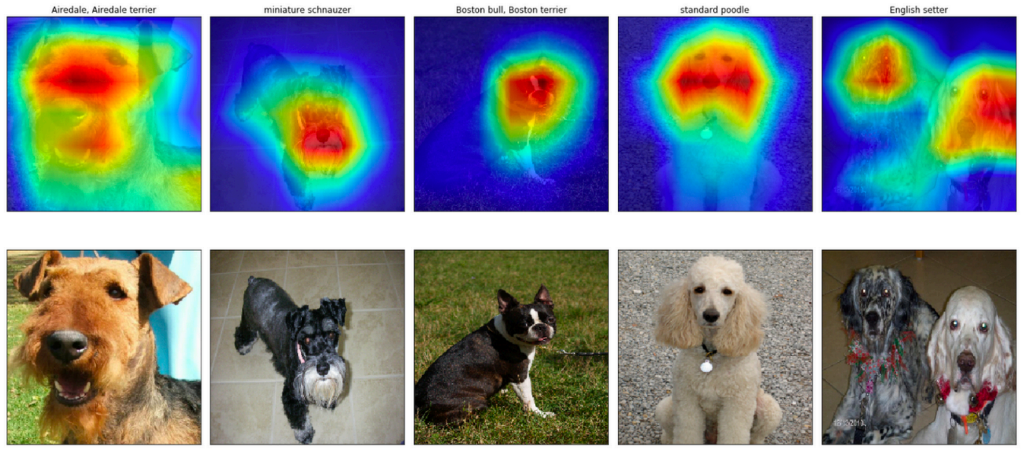


Deconv and guided back-propagation

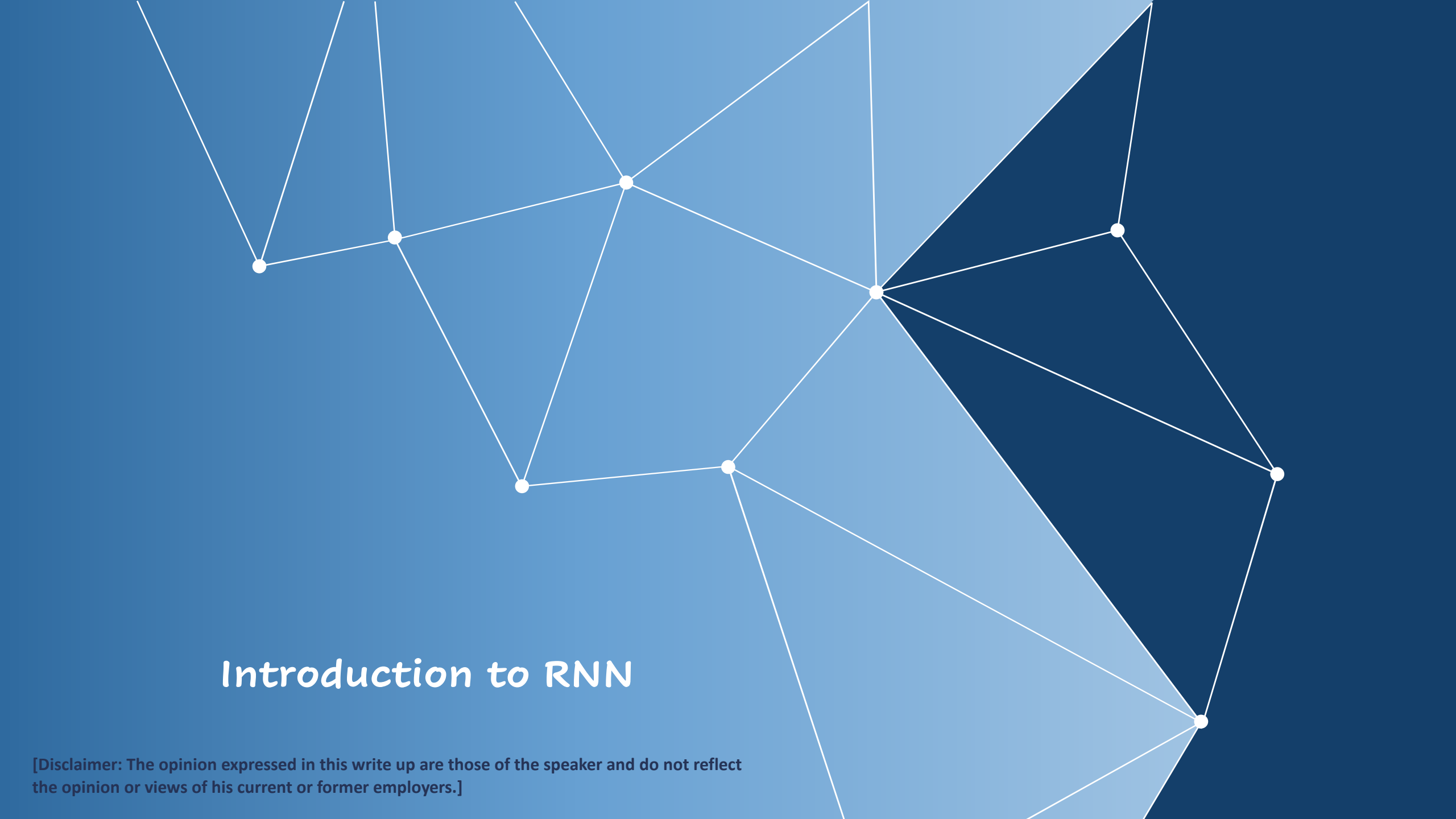


GB for both ”Cat” and “Dog”

Global average pooling for object localization



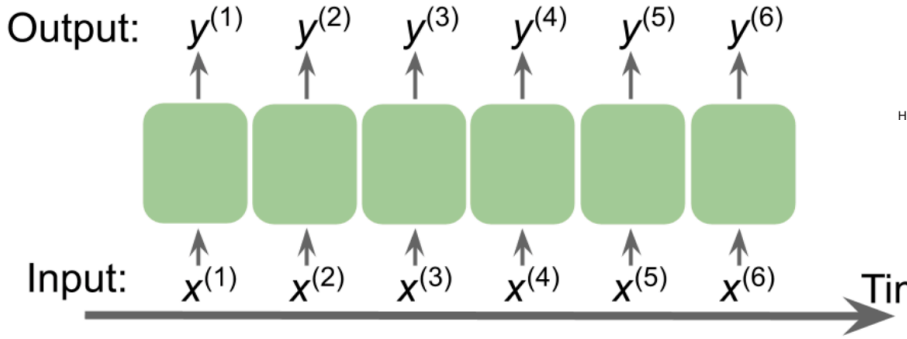
Source: Grad-CAM: Visual Explanations from Deep Networksvia Gradient-based Localizatio



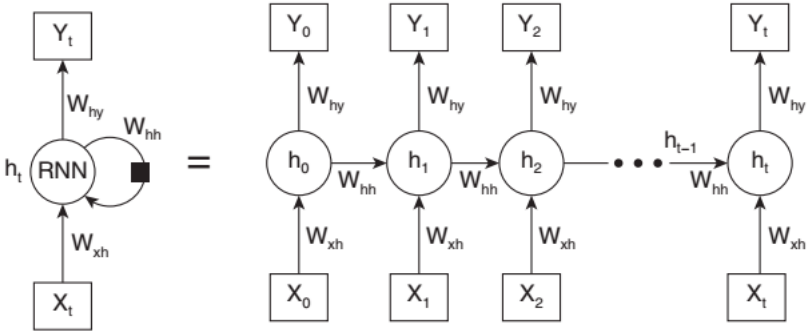
Introduction to RNN

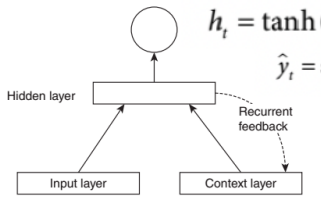
[Disclaimer: The opinion expressed in this write up are those of the speaker and do not reflect the opinion or views of his current or former employers.]

RNN – Recurrent Neural Network



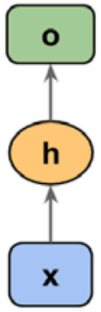
Example : Sequential Data



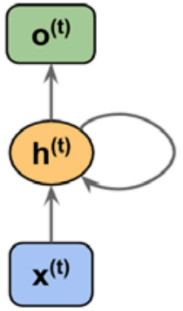


Elman Network

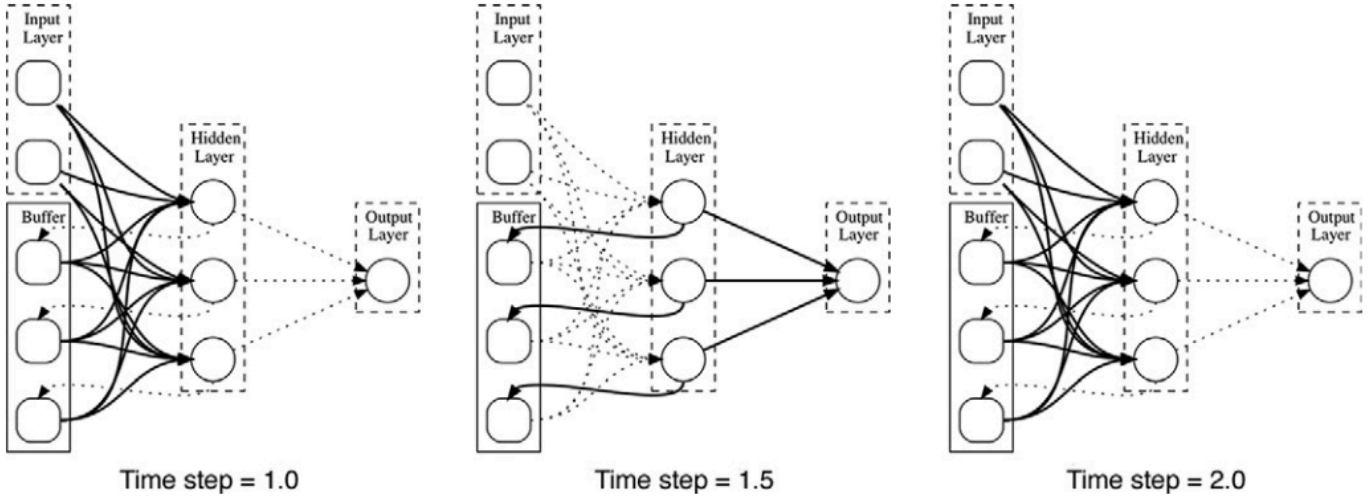
A standard feedforward network



Recurrent neural network



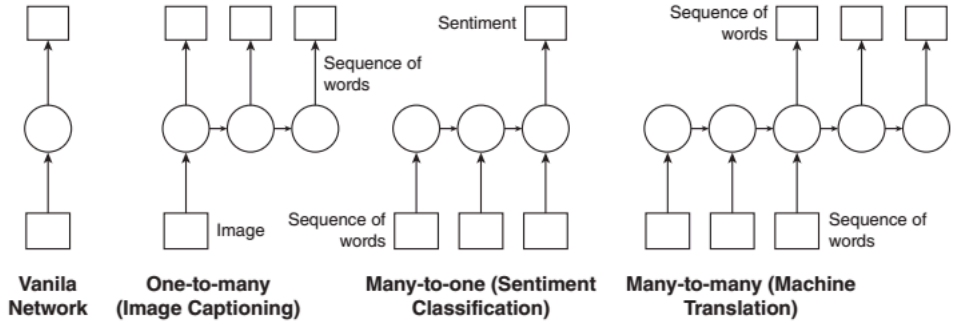
RNN ~ Looping Mechanism



Example : Recurrent Neural Network

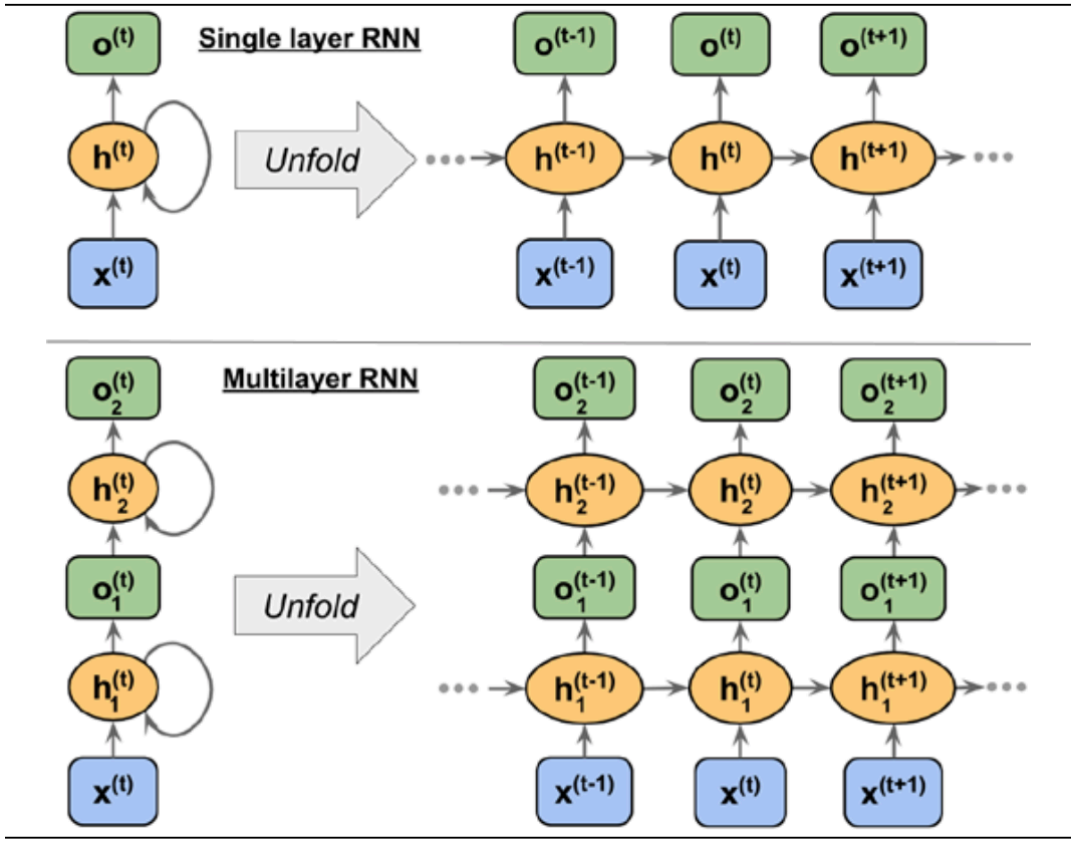
Source: "Python Machine Learning – 3rd Edition by Sebastian Ruschka

RNN – Various Architectures



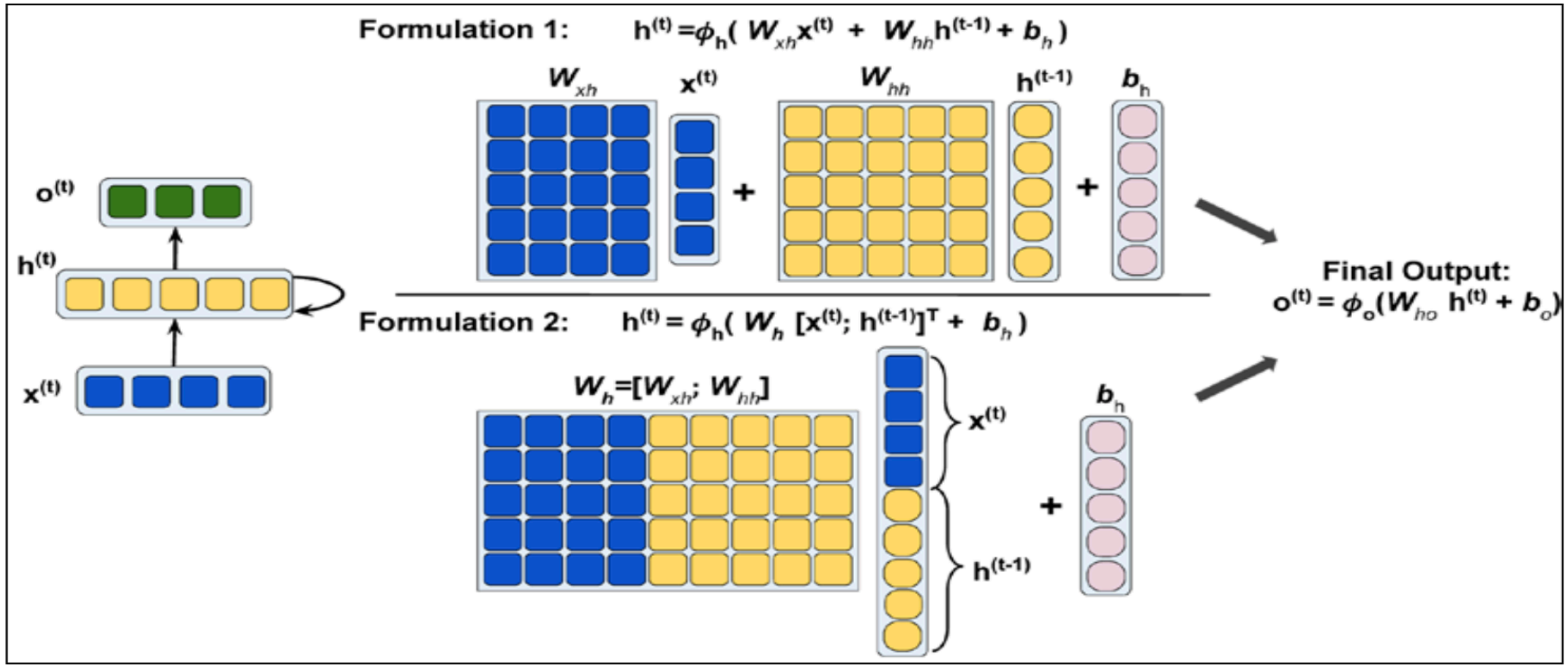
RNN – *can be arranged in various shapes of architectures:*

- **Vanilla Network**: A set of input and output vectors are connected through RNNs. Can be leveraged for stock market price prediction.
- **One-to-Many Network**: Here, input data is in standard format and output is a sequence. Mostly used for Image captioning.
- **Many-to-One Network**: The input data is a sequence, but the output is a fixed-size vector or scalar, not a sequence. Used in sentiment classification.
- **Many-to-Many Network**: Both the input and output arrays are sequences. An example of a synchronized many-to-many modeling task is video classification.



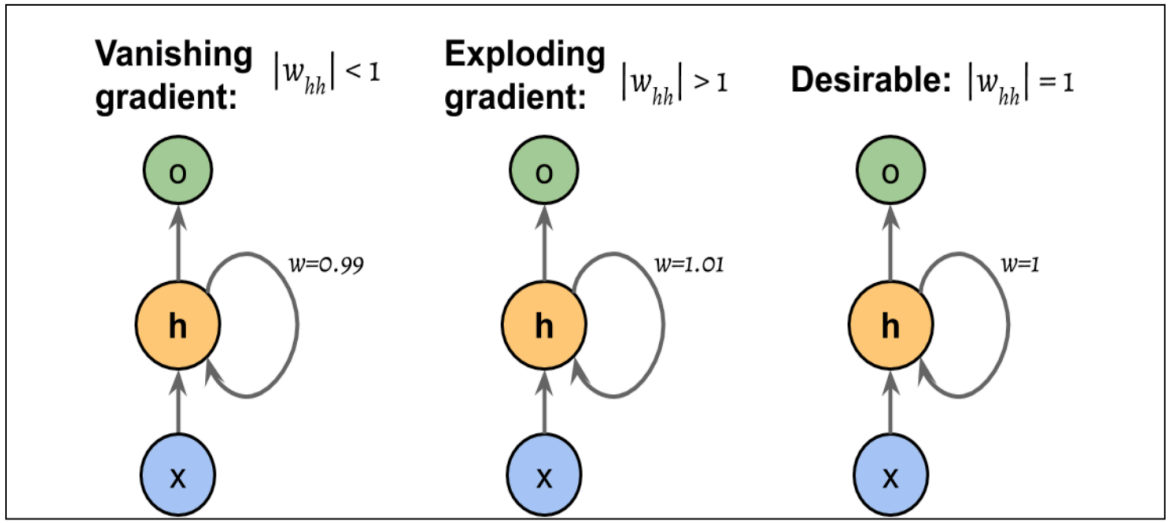
Unrolling an RNN

RNN – A glimpse of RNN flow



The process of computing the activations

RNN – Issues and Introduction to LSTM and GRU

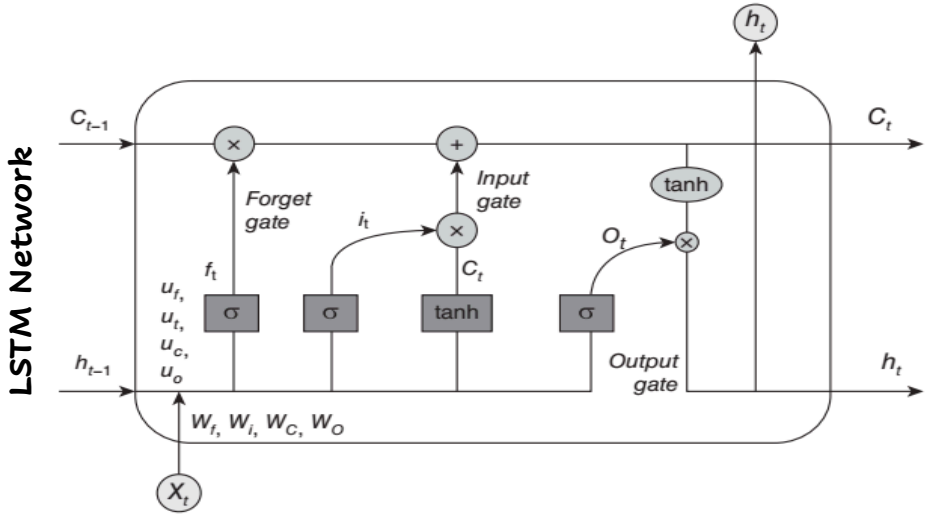


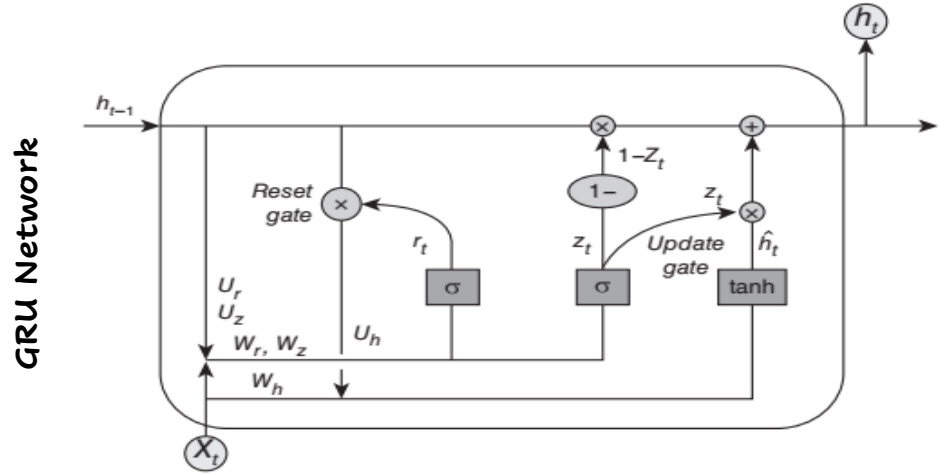
A few major issues with Vanilla RNN Network:

- **Vanishing Gradients**
- **Exploding Gradients**

Solutions:
- **Gradient Clipping**
- **TBPTT or Truncated Back Propagation through time**
- **LSTM – Long- Short Term Memory**

Source: "On the difficulty of training Recurrent Neural Networks





More on LSTM and GRU

In a **Long-Short Term Memory (LSTM)** Cell architecture, there are 3 different types of gates:

- **Forget Gate (f_t)**: The forget gate (f_t) allows the memory cell to reset the cell state without growing indefinitely. In fact, the forget gate decides which information is permitted to go through and which information to suppress. Now, f_t is computed as follows:

$$f_t = \sigma\left(W_{xf}x^{(t)} + W_{hf}h^{(t-1)} + b_f\right)$$

- **Input Gate:** It determines how much new information to be added to new cell state.

$$i_t = \sigma\left(W_{xi}x^{(t)} + W_{hi}h^{(t-1)} + b_i\right)$$

- **Output Gate:** It determines how much information needs to be passed to the output state.

$$o_t = \sigma\left(W_{xo}\,x^{(t)} + W_{ho}h^{(t-1)} + b_o\right)$$

- **Cell State:** It refers to the internal memory of the cell, which works like long-term memory of the cell.

$$\tilde{C}_t = \tanh\left(W_{xc}\,x^{(t)} + W_{hc}\,h^{(t-1)} + b_c\right)$$

- **Hidden State:** Output state of the cell.

$$h^{(t)} = o_t \odot \tanh\left(C^{(t)}\right)$$

Source: "Learning to Forget: Continual Prediction with LSTM"

Gated Recurrent Unit (GRU) is in many respect a close cousin of LSTM, almost similar in terms of its performance for dealing with vanishing gradients except for a few differences. GRU is computationally more efficient due to lesser number of parameters to train, but in terms of accuracy, the performance of both LSTMs and GRUs is very similar.

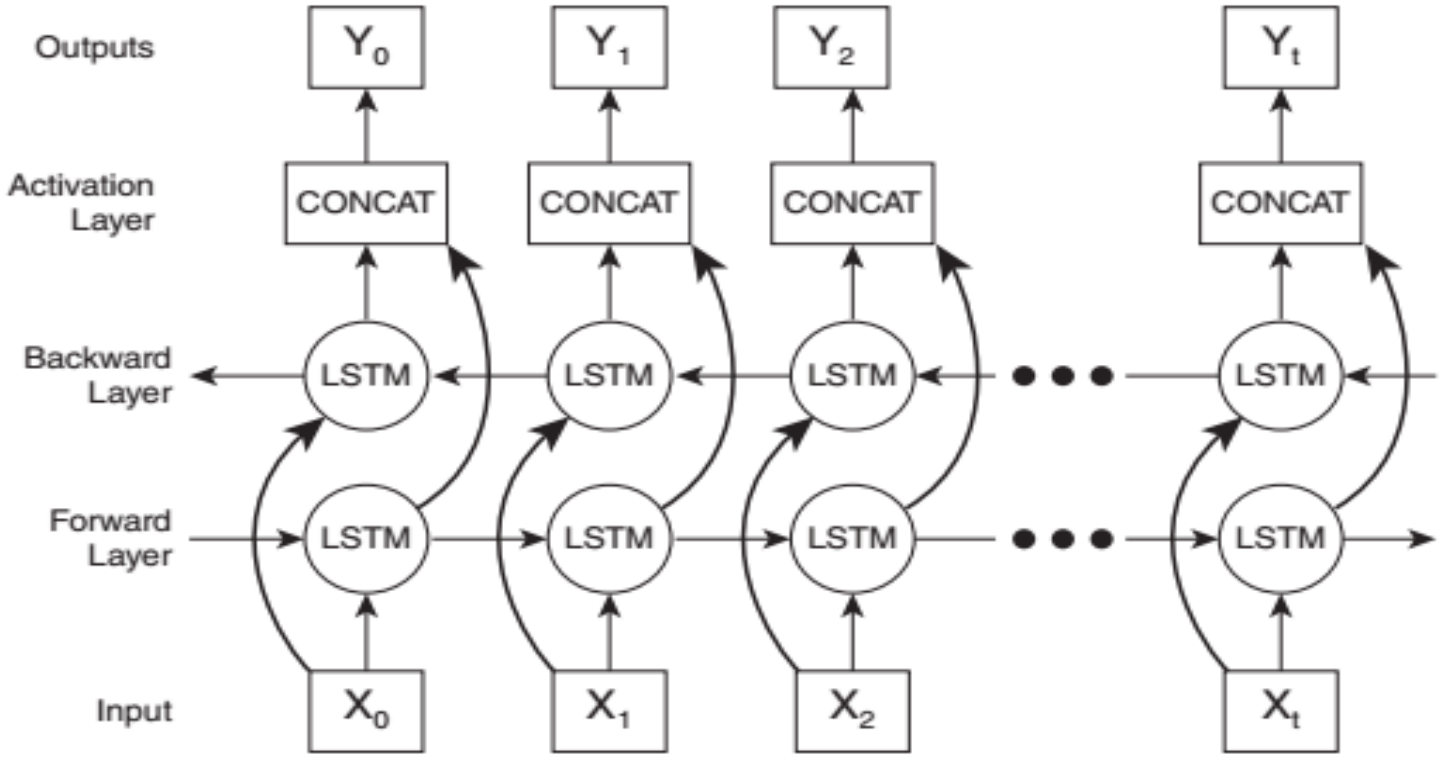
A few key characteristics of GRU network:

- **Reset Gate:** It basically determines how much past information to forget.
- **Update Gate:** It Determines how much proportion of past information to be retained and how much new information to be added. Of course, the past information used for computation

One important aspect about GRU is that there is no internal state memory in "GRU" and it has only a previous hidden state from which all the computations are being made.

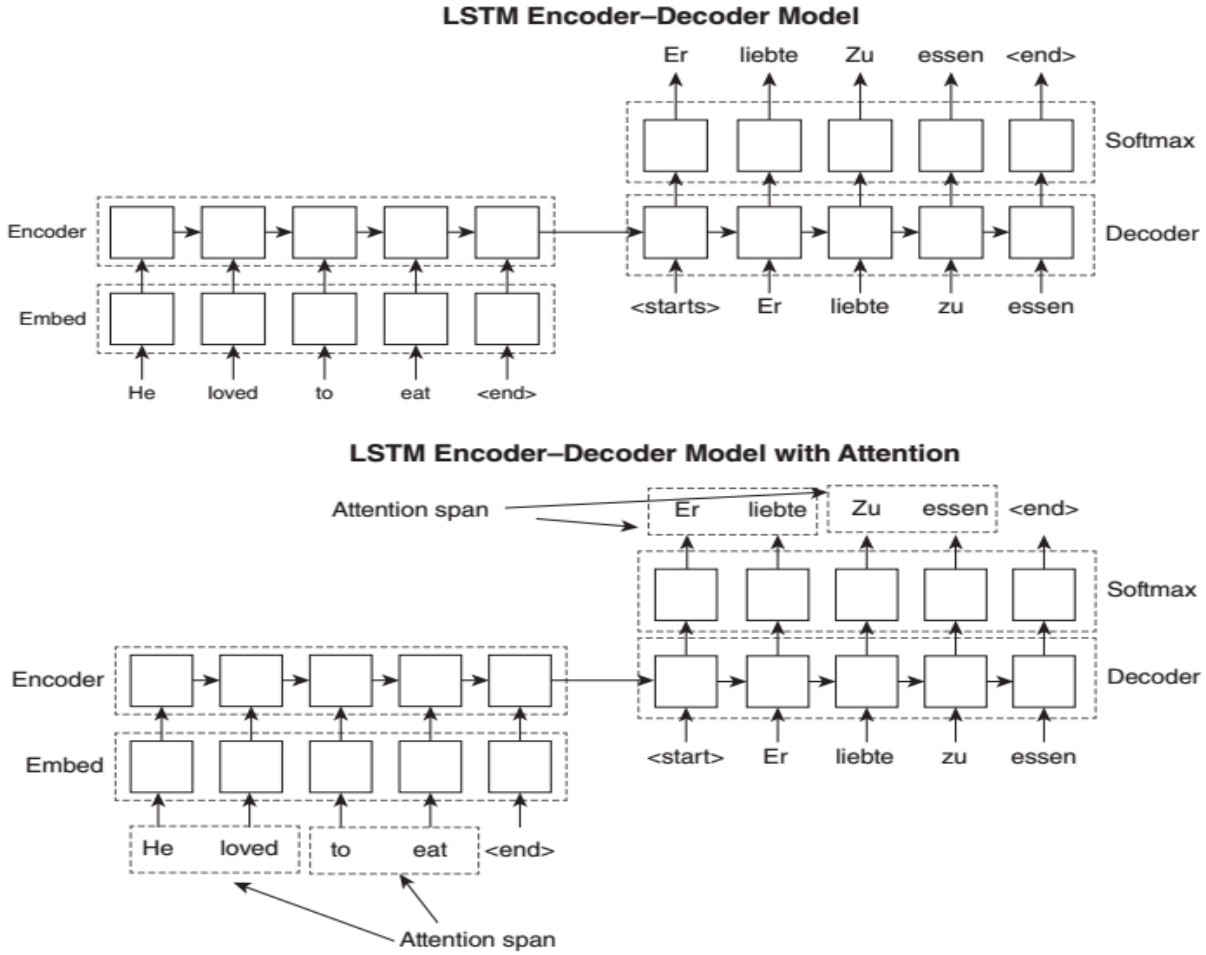
Source: "Empirical Evaluation ofGated Recurrent Neural Networkson Sequence Modeling"

Bidirectional LSTM



Source: "Bidirectional Recurrent Neural Networks"

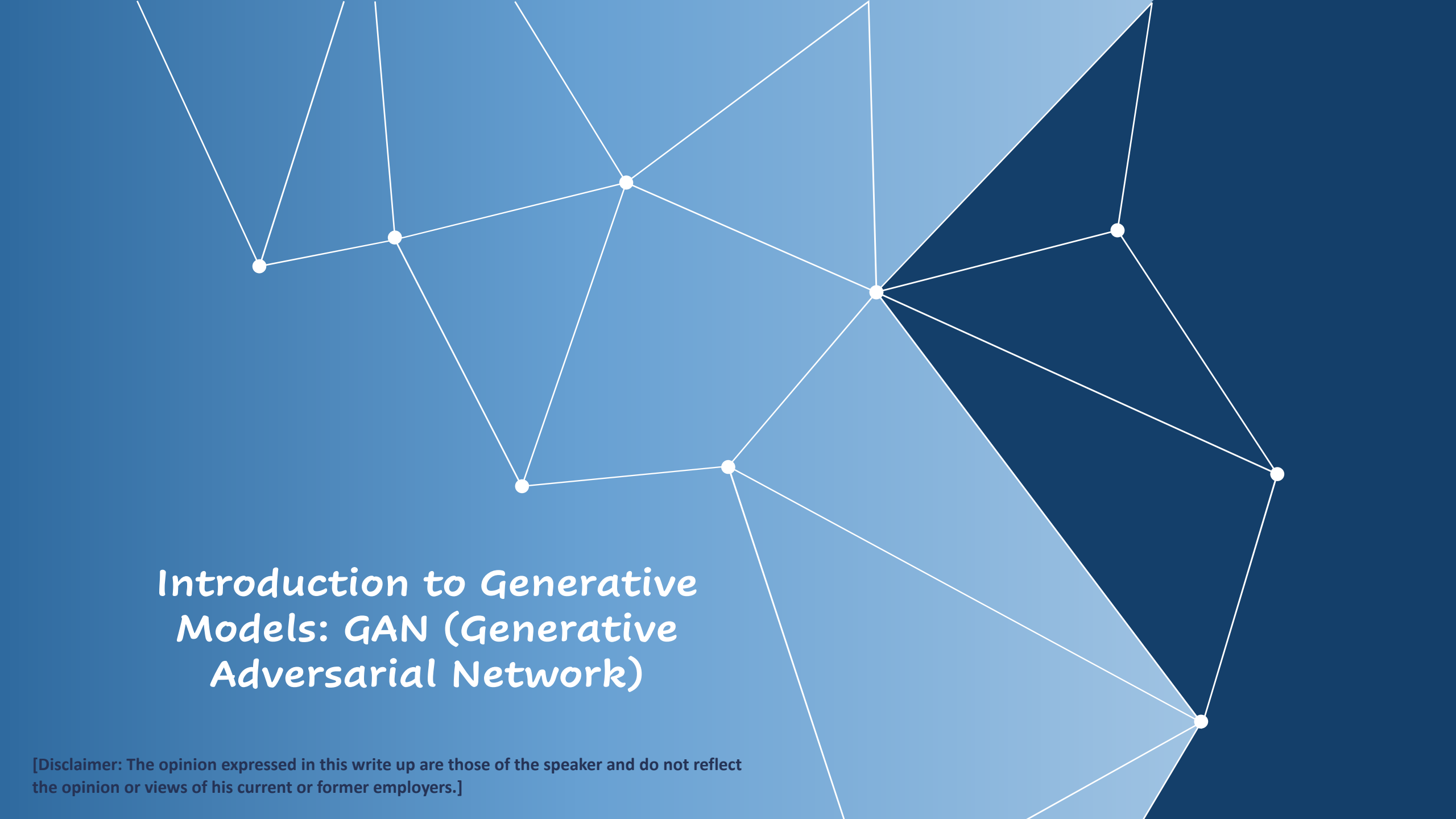
Attention Is All You Need!



Source: "Attention Is All You Need"

Case Study:

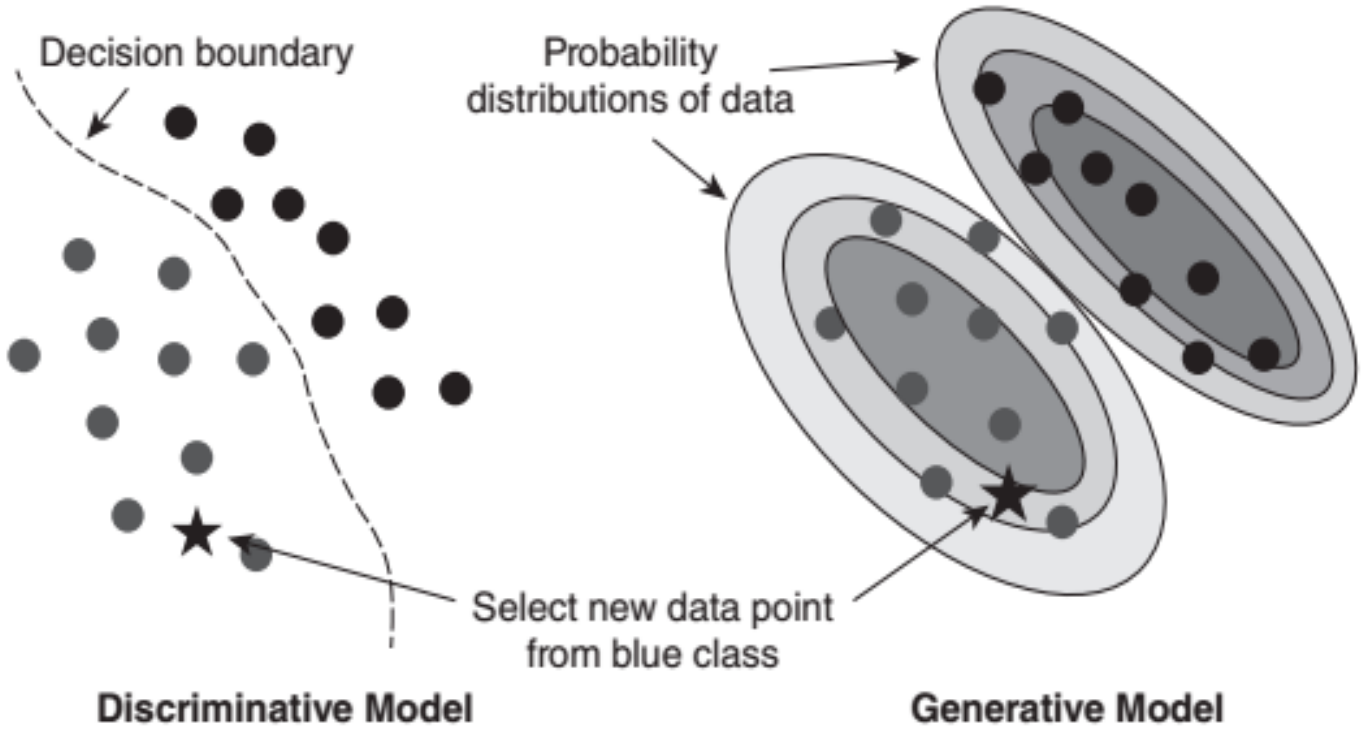
Application of RNN, RNN-LSTM and GRU → Demo



Introduction to Generative Models: GAN (Generative Adversarial Network)

[Disclaimer: The opinion expressed in this write up are those of the speaker and do not reflect the opinion or views of his current or former employers.]

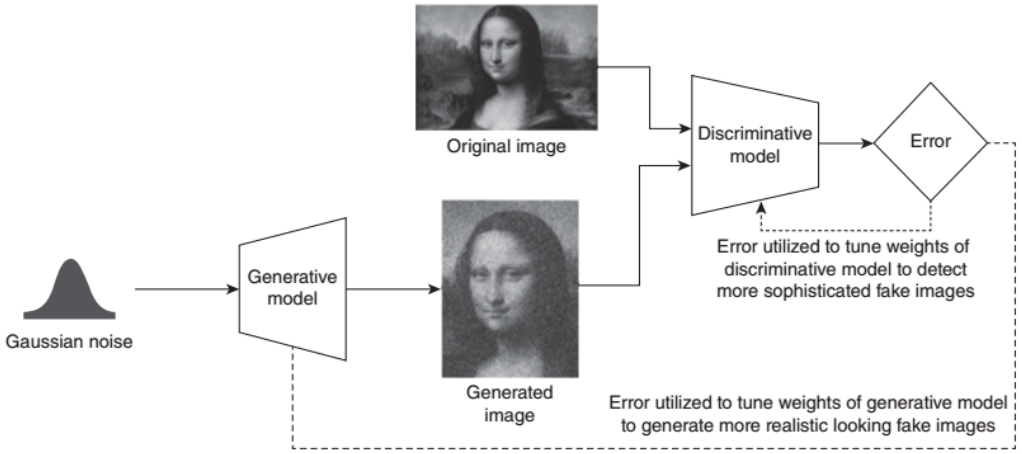
Discriminative Vs. Generative Models

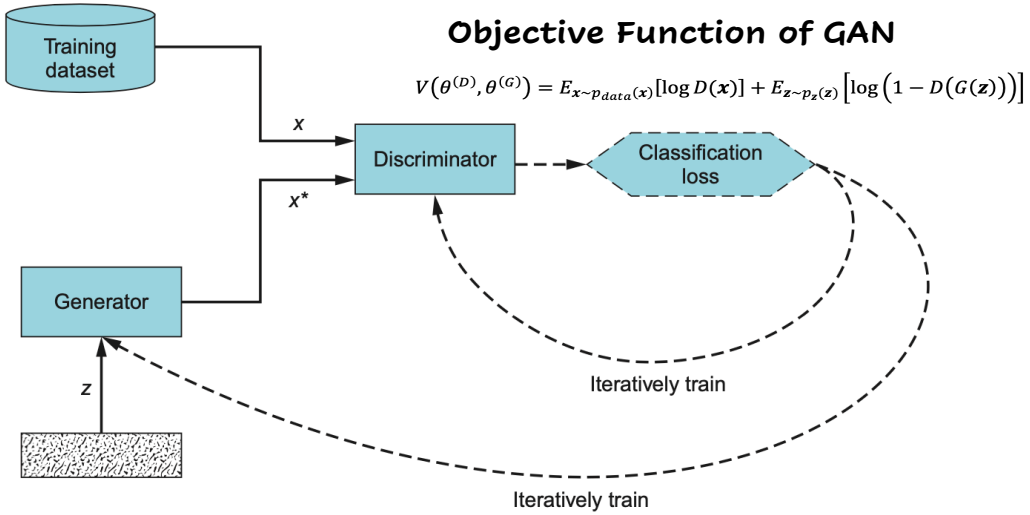


- For discriminative models, the objective is to create a boundary so that different classes can be separated. Such models can classify any point that falls into a particular class based on the region on which it has fallen.
- Typical classification and regression models are part of the discriminative models.
- "Generative models are traditionally defined as algorithms that aim to model data input distributions, p(**x**) or the joint distributions of the input data and associated target p(**x**, y)"
- "GANs are composed of two neural networks: a generator that tries to generate data that looks similar to the training data, and a discriminator that tries to tell real data from fake data. This architecture is very original in Deep Learning in that the generator and the discriminator compete against each other during training."

Source: "Generative Adversarial Nets"

Overview of GAN





- A GAN is an unsupervised learning algorithm , capable of generating new sample of data points unlike any deterministic models.
- A GAN is composed of basically two models, a generator and a discriminator.
- The task of generator model is to create various fake version of the sample data points which are almost quite similar to real data points. whereas the discriminator model's task is to distinguishingly identify whether a particular data point is real or fake.
- During the training phase, both the generative and discriminative models play a zero-sum game, in which one person's win is another person's loss and the profit and loss of both the players add up to zero. Due to this game, both models try to optimize their objectives separately, but nonetheless, both models use the error generated by discriminator model to tune the weights separately to win the game.

Case Study:

Application of GAN→ Demo