

Codebasics: Deep Learning with Tensorflow 2.0, Keras and Python

Why Deep Learning so Popular?

1. Data growth
 - a. Business, social media
2. Hardware advancements
 - a. GPU (Efficient for parallel computation)
 - b. TPU
3. Python & Open-source ecosystem
 - a. Pytorch, Tensorflow
4. Cloud & AI Boom

What is a neuron?

Given an age of a person, come up with a function that can predict if person will buy insurance or not. → Binary Classification

- a. Using linear regression?
 - But interrupted by outliers
 - Poor performance
- b. Using sigmoid or Logit (S-shaped) function
 - Sigmoid(z) = $1 / (1 + e^{-z})$, where e = Euler's number ~ 2.71828
 - Take an input value, converts into range 0 to 1
- c. Step by steps
 - Step 1: $Y = mx + c$ (linear)
 - Step 2: apply sigmoid $z = 1 / (1 + e^{-y})$

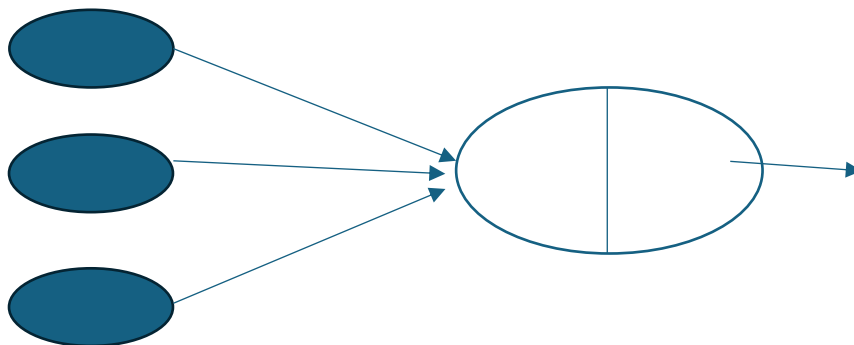
This is what **neuron** does! The neuron has a linear function and an activation function (e.g. sigmoid).

Logistic regression can be think as a single neuron!

Age => Feature

Insurance => Label / Target

$$y = \sum w^i x^i + b$$



Neural Network Simply Explained

Given an object e.g. Koala, a team of students (Mike, Mohan, Shakib) are trained to detect that specific object.

Each student is in charge of a single part (feature) of Koala face, e.g. eyes, nose, ears

Each student gives a probability score for that feature

After that, one coordinator (Serena) will collect the score from these three students

Serena will use its predefined formula to calculate if the object is Koala face or not.
(probability also)

Another team is formed to detect Koala body.

Each student (Yoti and Chen) is tasked to study the probability of each feature of Koala body

Another coordinator (Nidi) will integrate their findings to determine either it is Koala body (probability also)

Lastly, the head of teams, Sergey will collect the results from Serena and Nidhi, integrating their results to finally decide whether it is Koala. (Yes or No)

How NN is trained? **Backpropagation**

Give a random guess. Sergey will compare the predicted output with actual output with Supervisor. Knowing the result, Sergey will then tell Serena and Nidi about the **error feedback**, and subsequently Nidi and Serena will tell their team members.

The most fascinating about NN: Training / Learning process

Each neuron know how to learn the error and find the better way to solve it, given the labels.

PyTorch vs Tensorflow vs Keras

| PyTorch | Tensorflow | Keras |
|----------|---|---|
| Facebook | Google | Not a full-fledged framework, such as PT and TF |
| | With TF 2.0, keras is included / built inside | Nice wrapper : TF, CNTK, Theano |
| | | |

Neural Network for Handwritten Digits Classification

Refer Notebook: [7 Handwritten Digit Classification using ANN](#)

Notes

When all features is connected to each neuron in the hidden layer, it is called **Dense Layer**

Input = Each pixels (pixel value on grayscale image)

Output = 10 choices (0-9)

Activation Function

Refer Notebook: 8 Activation Function

Why Activation Function?

1. If not, neural network is just combined linear function
2. No need for hidden layers
3. However, we cannot solve problem with just linear functions
4. Previously, step function is introduced
5. But step function only output either 0 or 1
6. Hence sigmoid come to play

Other activation function

1. Tanh, from -1 to 1
2. Guideline: Use sigmoid in output later. All other places try to use tanh

Issue with sigmoid and tanh

1. Derivative at both end is almost 0
2. This create problems; if derivative very small, learning process cery slow → vanishing gradient

ReLU (x)

1. $\max(0, x)$
2. **Guideline:** for hidden layers, if you are not sure which activation function to use, just use ReLU as your default choice
3. Vanishing gradient still happen: if $x < 0$, derivative is 0

Leaky ReLU (x)

- $\max(0.1x, x)$

Note: No clear formula that which activation function is the best. Only through trial and error.

Derivatives ([Reference](#))

$$\text{Slope} = \Delta y / \Delta x$$

If $y = x^2$, slope = $2x$ (differentiation). But what does it mean?

- When $x = 2$, slope = $2(2) = 4$
- When $x = 5$, slope = $2(5) = 10$

Difference: Slope vs Derivative

| Slope | Derivative |
|--------------------------|------------------------------|
| Used for linear equation | Used for non-linear equation |
| Is a constant | Is a function |

Partial Derivatives ([Reference](#))

- $f(x, y) = x^3 + y^2$
- $\Delta f / \Delta x = 3x^2 + 0 = 3x^2$
- $\Delta f / \Delta y = 0 + 2y = 2y$
- With respect to which feature, just ignore other features

Why we need to know?

- $f(x, y) = x^3 + y^2$
- let $x = \text{bedrooms}$, $y = \text{sqr ft}$
- $\Delta f / \Delta x \rightarrow$ how much a **price** is changing given a change in **bedrooms**
- $\Delta f / \Delta y \rightarrow$ how much a **price** is changing in **sqr ft**

Neural Network is the game of **Adjusting Weights** ~

- Using partial derivative of errors to adjust individual weight, with respect to each feature

Matrix Basics

- Addition
- Subtraction
- Multiplication (Dot Product)

Refer Notebook

Loss or Cost Function

Refer Notebook: 11 Loss and Cost Function

Examples:

- 1 Sparse_categorical_crossentropy
- 2 Binary_crossentropy (log loss)
- 3 Categorical_crossentropy
- 4 Mean_absolute_error
- 5 Mean_squared_error

Cost Function

1. Calculate the loss of each instance
2. Calculate the total loss
3. Average the total loss → cost function

Epoch

1. Number of time that going through all training sample

For **logistic regression**, we use log loss / binary cross entropy

Gradient Descent for Neural Network

- Help to find the optimal **weights** and **bias**

Forward Pass

- Linear function
- Activation function
- Calculate error
- Repeat for all training samples (1 epoch)
- Calculate total error
- Calculate cost / loss function

Backward Pass

- Update the weights and bias
- $W1 = w1 - \text{learning rate} * \Delta \text{Loss} / \Delta w1$
- $W2 = w2 - \text{learning rate} * \Delta \text{Loss} / \Delta w2$
- $\text{Bias} = \text{bias} - \text{learning rate} * \Delta \text{Loss} / \Delta \text{bias}$

Forward Pass (Epoch = 2)

- Repeat the same process above
- This is based on batch gradient descent

Backward Pass

- Repeat the same as process above

**** Continue until your preset number of iterations (epoch)**

Stochastic Vs Batch vs Mini-batch

Batch

1. We go through all training samples and calculate cumulative error
2. Now we back propagate and adjust weights
3. Good for small training set

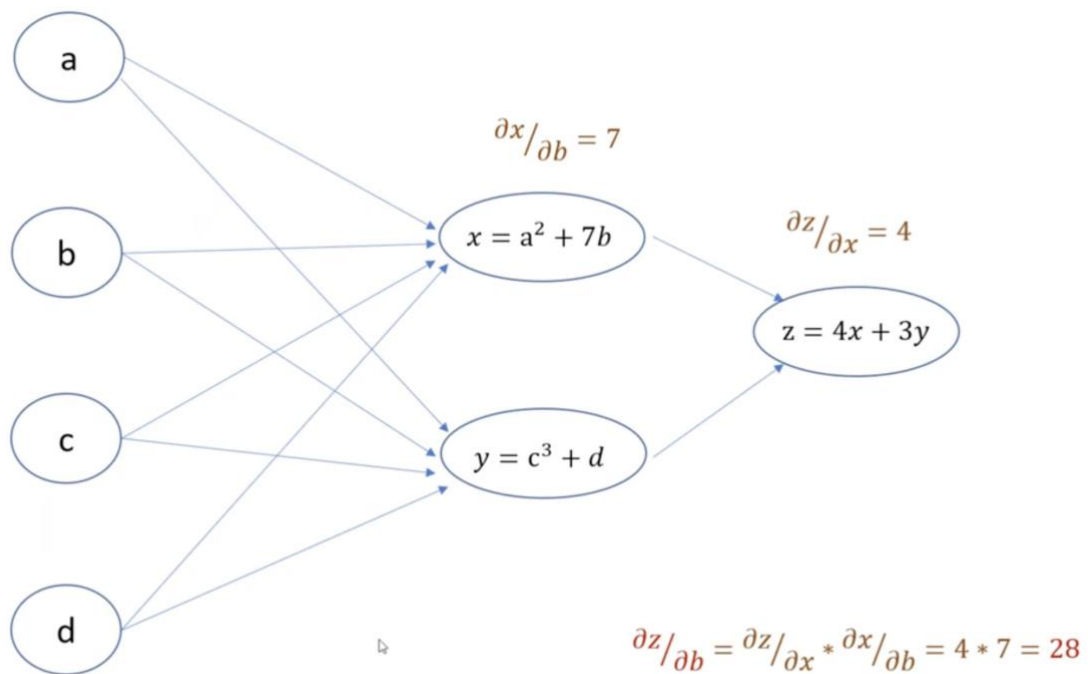
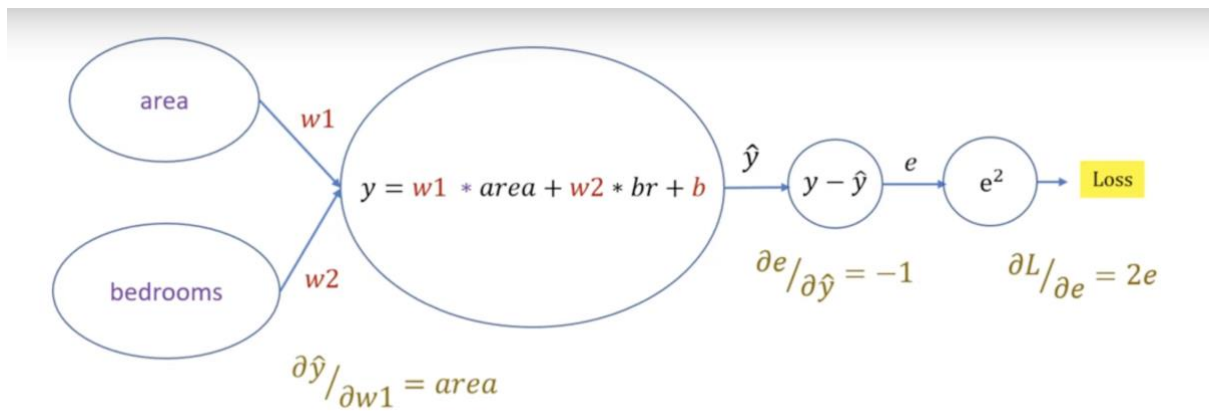
Stochastic

1. Randomly pick one sample
2. Adjust weights
3. Repeat again and again
4. Good when training set is big, we don't want too much computation

Mini-Batch

1. Randomly pick a batch of sample
2. Adjust weights
3. Repeat again and again

Chain rule



Dropout Regularization

Analogy:

When you buy shirt, you will not buy very fit shirt or very loose shirt. You will find the one that make you comfortable

Represent → Overfitting, Underfitting

Underfitting → too simple, no extract underlying patterns

Overfitting → too good to be true,

Dropout Regularization

→ randomly drop some neurons for each training sample

→ when dropping certain neurons, the network cannot rely on the certain inputs; will not create bias

→ will not learn redundant details of neurons

Handling Imbalance Datasets

2 Techniques

1. Under sampling majority class
 - 99000 Negative, 1000 Positive
 - Choose 1000 Negative randomly, and with 1000 Positive
2. Over sampling minority class by duplication
 - Simply copying minority class instances
 - Generate new sample from current sample by simply duplicating them
3. Over sampling minority class using SMOTE
 - Generate synthetic examples using k nearest neighbors algo
 - SMOTE → Synthesis Minority Over-Sampling Technique
4. Ensemble Method
 - 3000 Negative 1000 Positive
 - Split into 3 batches → 1000 Negative 1000 Positive (Resued)
 - See majority vote
5. Focal loss
 - Penalize majority class, give more weightage to minority class

Examples of imbalance datasets

1. Customer churn rate
2. Device failure
3. Cancer prediction

Application of Computer Vision

1. Image Classification & Object Detection
 - a. Google lens
 - b. Phone album
2. Banking field
 - a. Cheque OCR
3. Agriculture
 - a. Cambridge
4. Autonomous Car
5. Retails
 - a. Amazon go

Convolutional Neural Network

How to detect local features

- Using filter / kernel → feature detector

Go through convolution

- Element wise multiplication between image and filter
- Stride and padding
- Final product → Feature. Map

Feature extraction + Classification

Full Convolution Flow

1. Convolution Layer with ReLU → to create feature maps
2. Pooling → reduce dimension, computation, overfitting, model is more variant, E.g. max pooling, average pooling
3. Repeated of Convolution Layer with ReLU and Pooling
4. Flattening
5. Fully Connected

Benefits

1. Convolution
 - a. Connections sparsity reduces overfitting
 - b. Conv + Pooling gives location invariant feature detection
 - c. Parameter sharing
2. ReLU
 - a. Introduce non-linearity
 - b. Speeds up training, faster to compute
3. Pooling
 - a. Reduce dimension
 - b. Reduce overfitting
 - c. Model more tolerant to variation

Limitation

1. CNN does not take care of rotation and scale

CNN will learn the filter automatically, just like the weights in ANN.

Convolution: Padding and Stride

When we apply filter (e.g 3x3) to the image, we actually reduce a bit dimension

- E.g. Image (5x7, mxn), filter (3x3, fxf)
- E.g. Feature map (3x5) → calculated from $(m-f+1) \times (n-f+1)$
- Valid convolution
- Disadvantage: corner pixels don't contribute as much in feature detection

How to solve the problem? **Padding** ~

- Pad zero on 4 sides on input image → from (5x7) to (7x9)
- After convolution via kernel (3x3) → get back original image dimension (5x7)
- Same convolution

Stride (1,1) → move right 1, down 1 per step

Stride (2,2) → move right 2, down 2 per step

padding, stride = hyperparameter to our models ~~

Data Augmentation to Address Overfitting

CNN cannot solve rotated image

- Can use data augmentation
- Create different kinds of transformation to create new instances with variant scaling and contrast for example.

Transfer Learning

1. Using pre-trained model
2. Using their trained weights and bias
3. Using frozen layers
4. Add on or modify last few layers to suit your problem.

Wikipedia Definition

→ TL is a research problem in ML that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks.

Mobilenet V2 Model

- Trained at Google
- 1.4 mil images
- 1000 classes (more general)

Object Detection vs Image Segmentation

Object Detection → draw the bounding boxes

Image Segmentation → classify each of the pixel to classes (such as masking)

3 Popular Datasets for CV

1. ImageNet
 - a. 14-million hand annotated images
 - b. 1 annotated bounding box for at least 1 million images
 - c. Annotated using crowdsourcing
2. Coco
3. Google Open

Sliding Window Object Detection

A small window is used rather than the whole image.

Slide through the image to detect the object

Find the location

How to determine the size of window? Trial and Errors

Disadvantages → too much computation, bounding box is not accurate

To solve these issue → R CNN → Fast R CNN → Faster R CNN → YOLO

- Try window in reduced region

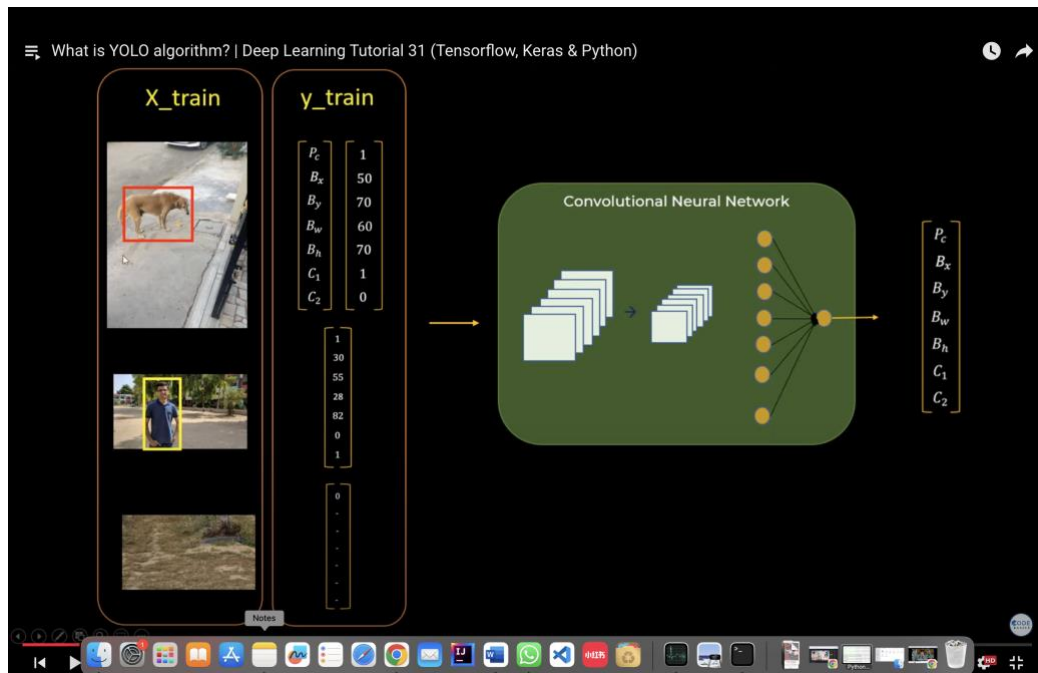
YOLO Algorithm – You Only Look Once

Image Classification

- Is this a dog or a person
- Dog = 1, Person = 0

Object Localization

- Where exactly is the dog in this image?
- Dog = 1, Person=0 with Bounding box



- However, this works ok only for single object. What about multiple objects in an image?

YOLO

- Separate the image into grid cells (nxn)
- Detect the center of object

Recurrent Neural Network

Sequence Modeling

1. Google Gmail auto- complete
2. Google Translation – Machine Translation
3. Named Entity Recognition
4. Sentiment Analysis

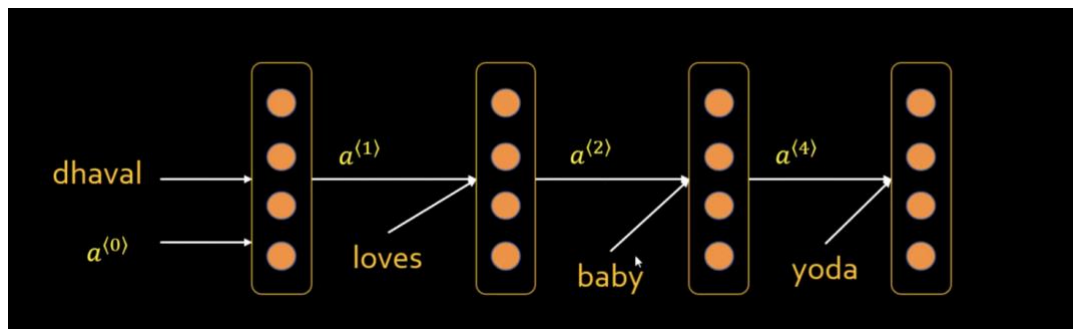
Sequence is important ~

Why not simple ANN?

- Variable size of input/output neurons
- Too much computation
- Parameters not shared

Named Entity Recognition

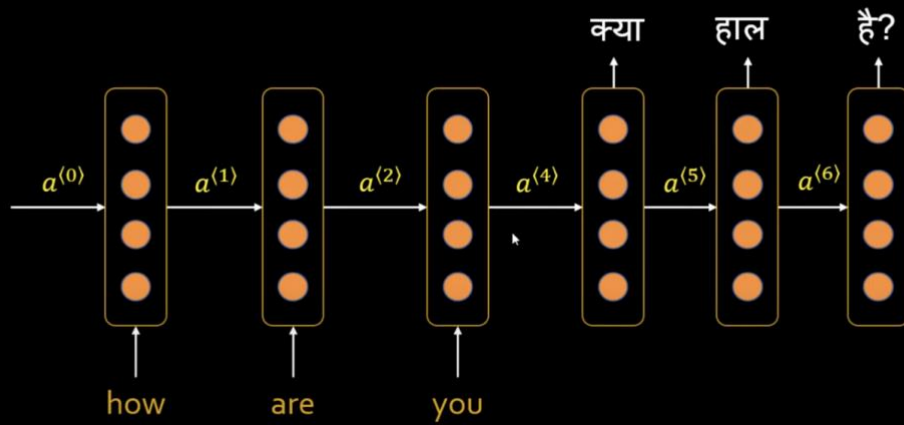
- Recognized pre-defined entities
- E.g. of how RNN works
- Given sentence Dhaval loves baby yoda.



-
- Only one network is required.

Language translation

how are you? क्या हाल है?



Types of RNN

1. Many to Many
 - a. Named Entity Recognition
 - b. Language Translation
2. Many to One
 - a. Sentiment Analysis
3. One to Many
 - a. Music Generation → feeding one starting tone/node

Vanishing / Exploding Gradient

Vanishing

- When deeper network, when updating weights, weights in the first few layers are hardly learned. (hardly converge)

Exploding

- Same as vanishing, but weights would change drastically due to huge value (easily diverge)

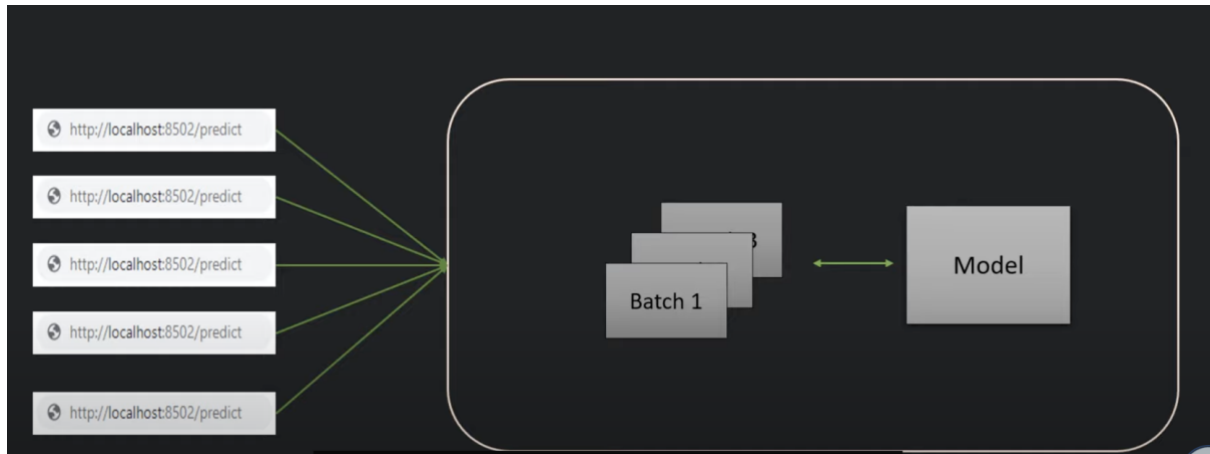
How to solve?

- GRU
- LSTM

Tf serving tutorial

- Most of the case, we use flask or fastapi to run backend
- We need to create several model (alpha, beta version)
- With tf serving, make model management and model serving very easy

Batching Inferences



Using tf serving

- You no need manually write backend server files
- You can run various version (v1, v2, v3)
- You can name various version (production, beta ...)

How

- Use docker
- Operate using docker
- Prepare config files for each functions

Quantization

In most of the cases, we want to deploy the model to the edge devices. But edge devices have a very limited memory to run this model. To solve this , we use quantization.

- A process of reducing model size so that it can run on edge devices (e.g. handphones, smart watch)
- Convert the weights (data type: float 64) into efficient data types (float 16)

Benefits

- Run ML models efficiently in edge devices
- Faster inference

2 ways to perform

1. Post training quantization
 - a. Tf.lite.convert
2. Quantization aware training
 - a. Quantized model
 - b. Fine-tuned quantized model
 - c. Tf lite convert
 - d. – More work, but high accuracy

https://www.tensorflow.org/model_optimization/guide/quantization/training

https://www.tensorflow.org/model_optimization/guide/quantization/post_training

<https://blog.tensorflow.org/2020/04/quantization-aware-training-with-tensorflow-model-optimization-toolkit.html>

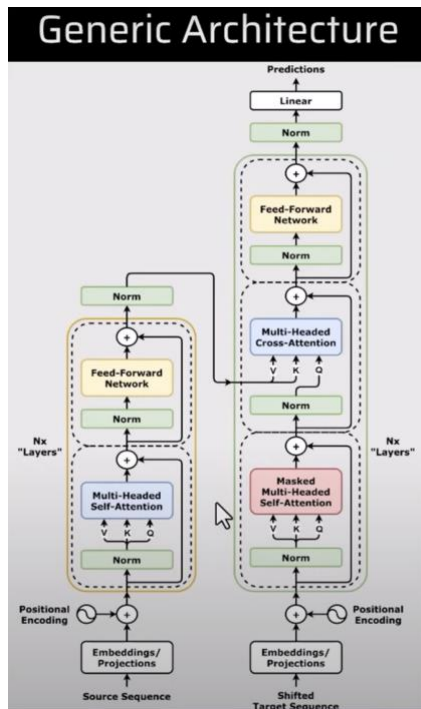
Transformers

Language Model – BERT, GPT

Large LM – GPT (Trained on humongous amount of data)

Topics you need to know

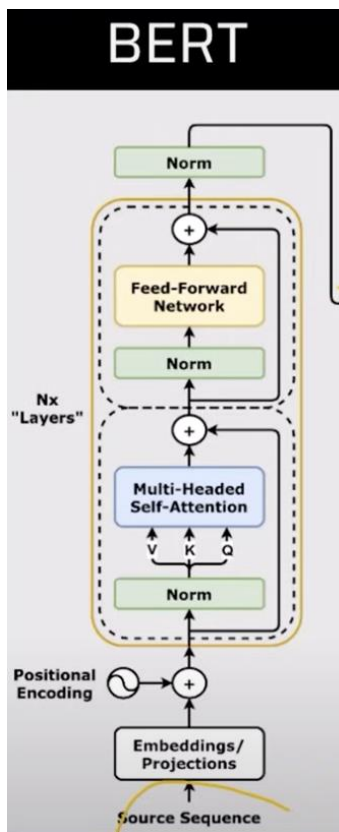
1. Word Embeddings (Static vs Contextual)
 - a. Machine don't know text, so need to represent text in term of number
 - b. Feature vector? But too many features and no variant
 - c. Use learned weights from studying a fake problem via neural network → Word2Vec. All the weights are the features of the individual word
 - d. The distance between King and Queen → gender direction
 - e. Study humongous to capture the relationship
 - f. Limitations of static embeddings
 - i. No contextual relationship → same word might have more than one meaning, cannot capture different contextual meaning
 - g. Solution? Use contextual embeddings
 - i. Dish – Rice dish – Indian rice dish --- Sweet Indian rice dish
 1. Capture the words that influence the static embeddings as features
2. Transformer Architecture – Encoder + Decoder
 - a. Encoder
 - i. Create contextual embeddings
 - b. Decoder
 - i. Predict the next word (e.g.) using the contextual embeddings
 - ii. Inputs: contextual embeddings + previous predicted words
 - iii. Other task: language translation



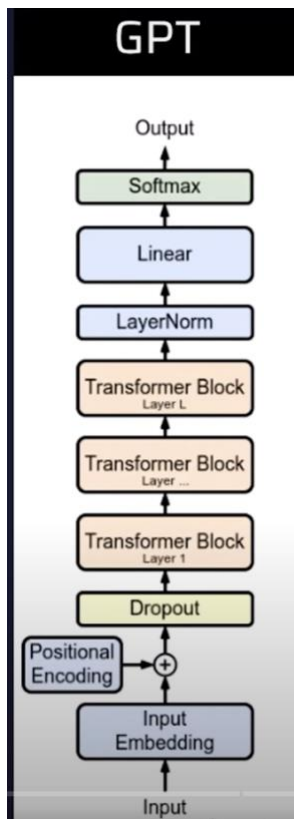
c. Encoder + Decoder

2 Stages \rightarrow Training + Inference

BERT \rightarrow only encoder part, 768d



GPT → only decoder part, 12258 d



W_q, W_k, W_v . → talk later

Inside the encoder (Parallel computing)

1. Tokenize the sentence
2. Get static embeddings + positional embeddings
3. Attention – capture contextual words that attend to / enrich the current predicted word
4. Query Key Value – modifier of the predicted / attended word (sweetness, indianness, riceness)
5. Add the value to the static + positional embeddings → forming contextual aware embedding
6. Query = $W_q \times E$; where W_q knows how to encode 'query' of a token for attention computation
7. Key = $W_k \times E$; where W_k knows how to encode 'key' of a token for attention computation
8. Query x Key → go through softmax → Weight (probability)

9. $\text{Value} = W_v \times E$; where W_v knows how to encode 'value' of a token for attention computation
10. $\text{Context} = \text{Value} \times \text{Weight}$ (from step 8)

Multi Head Attention

1. 1 head working on adjective, another head working on verb, other might work on pronoun
2. Focus on different aspects or types of relationships (semantic, positional, syntactic) simultaneously, enriching the contextual understanding of each token

Decoder

- Predict the next word based on each token
- Use Query from decoder, Key and Value from encoder