Advanced Machine Learning: Data Science and ML Refresher

Shubhankar Agrawal

Abstract

This document serves as a quick refresher for Data Science and Machine Learning interviews. It covers mathematical and technical concepts across a range of algorithms. This requires the reader to have a foundational level knowledge with tertiary education in the field. This PDF contains material for revision over key concepts that are tested in interviews.

Contents

1. Deep Learning

1.1. Key Concepts

Neural Networks - Architecture similar to neurons

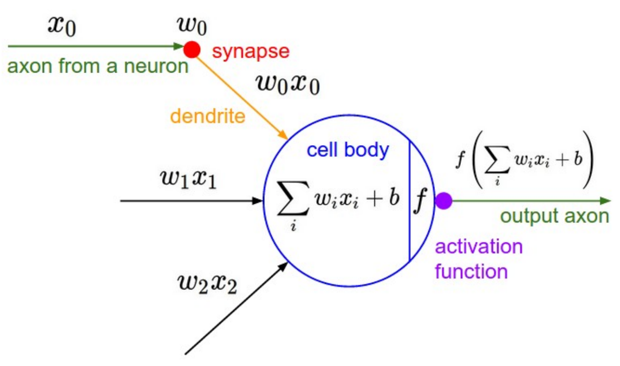


Figure 1. Neural Networks
[2]

1.1.1. Terminology

Commonly used terms

Model: Function $y = f(x)$ parametrized by θ
Epoch: One full pass of the training set
Batch: Subset of the training set
Back-propagation: Weigh updates with gradients

Table 1. Gradient Descent

Name	Samples	Pros	Cons
Batch	All	Stable Convergence	Memory, Slow
Mini-Batch	Subset	Efficient, Parallel	Tune Size
Stochastic	1	Escape local minima	Noisy

Layer sizes commonly used 2^n, optimized for GPU computations and memory allocations

1.1.2. Regularization

Common methods:

- L1 - Lasso
- L2 - Ridge
- Data Augmentation (Scale, Rotate, Flip, Noise)

Dropout

- Train: Randomly set neurons to 0 (probability p)
- Infer: Scale activations by $1 - p$

Early Stopping: Stop when validation loss plateaus
Batch Normalization: Stabilize training, convergence

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{1}$$

1.2. More Components

1.2.1. Activations

Introduce Non-Linearity, summarized in Figure 5.
Notes:

- ReLU / Leaky ReLU usually used in between layers
- ReLU can cause dead neurons (Solve with Leaky)
- Tanh used to centre outputs around 0
- Sigmoid for Binary output
- Softmax for multiclass output

Vanishing Gradient: Activation derivative tends to 0. Causes neurons to turn off. Switch to other activations (ReLU).
Exploding Gradient: Gradient becomes too large while propagating. De-stabilizies convergence. Use gradient clipping, Batch Normalization.
Gradient Checking: Check analytical (Back-propagated) vs Approximate gradient. The smaller the value the better

$$\begin{aligned} \nabla_\theta J(\theta) &\approx \frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon} \\ \text{Difference} &= \frac{||\nabla_\theta J_{\text{analytical}} - \nabla_\theta J_{\text{approximate}}||}{||\nabla_\theta J_{\text{analytical}}|| + ||\nabla_\theta J_{\text{approximate}}||} \end{aligned} \tag{2}$$

NOTE: Batch Normalization should be applied after ReLU otherwise the normalization is lost after the activation

1.2.2. Optimizers

Gradient Descent

$$\theta_t = \theta_{t-1} - \eta \nabla_\theta J(\theta_{t-1}) \tag{3}$$

Momentum
Exponential decay on moving average of gradients

$$\begin{aligned} v_t &= \beta v_{t-1} + (1-\beta)\nabla_\theta J(\theta_{t-1}) \\ \theta_t &= \theta_{t-1} - \eta v_t \end{aligned} \tag{4}$$

AdaGrad
Scale by inverse square root of running average of squared gradients.

$$\begin{aligned} s_t &= s_{t-1} + (\nabla_\theta J(\theta_{t-1}))^2 \\ \theta_t &= \theta_{t-1} - \frac{\eta}{\sqrt{s_t + \epsilon}} \nabla_\theta J(\theta_{t-1}) \end{aligned} \tag{5}$$

RMSProp
Improves Adagrad with a Decay factor to prevent fast diminishing weights

$$\begin{aligned} s_t &= \beta s_{t-1} + (1-\beta)(\nabla_\theta J(\theta_{t-1}))^2 \\ \theta_t &= \theta_{t-1} - \frac{\eta}{\sqrt{s_t + \epsilon}} \nabla_\theta J(\theta_{t-1}) \end{aligned} \tag{6}$$

Adam (Adaptive Moment Estimation)
Combines Momentum and RMSProp

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta_{t-1}) \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta J(\theta_{t-1}))^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \\ \theta_t &= \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t \end{aligned} \tag{7}$$

1.2.3. Schedulers

Adapt learning rate over epochs for faster convergence and optimized search

Table 2. Learning Rate Schedulers

Method	Description
StepLR	Reduce by constant factor n iterations
ExponentialLR	Reduce exponentially
CyclicLR	Cycle from base to minimum
ReduceLROnPlateau	Reduce when the metric plateaus

1.2.4. Code

```
model.train(True)
optimizer.zero_grad()
outputs = model(inputs)
loss = loss_fn(outputs, labels)

loss.backward()
optimizer.step()

model.eval()
with torch.no_grad():
    outputs = model(inputs)
```

Code 1. PyTorch Model

1.3. Convolutional Neural Networks

Modelling spatial features.
Input Dimensions: $(BatchSizexHeightPixelsxWidthPixels)$

$$(I * K)(x,y) = \sum_m \sum_n I(x+m, y+n) \cdot K(m,n) \tag{8}$$

Table 3. CNN Layers

Layer	Description	Purpose
Convolutional	Kernel Multiplication	Spatial Features
Pooling	Aggregates (Max / Avg)	Down Sample
Fully Connected	Weight Multiplication	Learning

$$\text{Output Size} = \frac{(W - K + 2P)}{S} + 1 \tag{9}$$

where:

- W is the input size (width or height),
- K is the kernel size,
- P is the padding applied,
- S is the stride of the convolution.

ResNet
CNN with skip connections (Concat identity + convolution features). Solves vanishing gradient.

1.4. Recurrent Neural Networks

Modelling sequential (or) temporal features.
Input Dimensions: $(BatchSizexFeaturesxTimeSteps)$

- Maintains a hidden state h_t to store temporal influence

$$h_t = f(W_h \cdot h_{t-1} + W_x \cdot x_t + b) \tag{10}$$

Batch Learning

- Process encoding in parallel
- Hidden state updates sequentially

Truncated Back Propagation Through Time (BPTT): Limit propagating gradients through full sequence. Prevent Vanishing Gradients.
Inference: No teacher forcing, i.e. use $\hat{y}_t$ as inputs x_{t+1}

1.4.1. LSTM

Long Short Term Memory

- Two hidden states (Short Term, Long Term)
- Three gates (Input, Forget, Output)

$$\begin{aligned} \text{Forget } f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ \text{Input } i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \text{Cell (Long) } C_t &= f_t \cdot C_{t-1} + i_t \cdot \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ \text{Output } o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ \text{Hidden (Short) } h_t &= o_t \cdot \tanh(C_t) \end{aligned} \tag{11}$$

1.4.2. GRU

Gated Recurrent Unit: Simplied version of LSTM with fewer parameters

- Single hidden state
- Two gates (Update and Reset)

$$
\begin{aligned}
\text{Update } z_t &= \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \\
\text{Reset } r_t &= \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \\
\text{Hidden } h_t &= z_t \cdot h_{t-1} \\
&\quad + (1 - z_t) \cdot \tanh(W_h \cdot [r_t \cdot h_{t-1}, x_t] + b_h)
\end{aligned} \tag{12}
$$

RNN architectures summarized in Figure 6

1.4.3. Seq 2 Seq

Sequence to sequence models: Input Sequence -> Output Sequence

1.5. Transformers

Seq 2 Seq models with an **Encoder** and **Decoder** architecture.

- Replace RNNs with self-attention
- Process sequences in parallel

1.5.1. Attention

Identifies relevant parts of sequence for each token

Scaled Dot-Product Attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{13}$$

where Q: Query, K: Key, V: Value, d_k: Dimensions

Multi-Head Attention:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{14}$$

- Computes attention over multiple independent heads
- Each head learns unique representations

Types of Attention

- Self-Attention: Within the same sequence
- Cross-Attention: Between encoder and decoder sequences
- Masked Self-Attention: Used in autoregressive tasks (e.g., GPT) to prevent looking at future tokens

Input Representations

- **Token Embeddings:** Maps tokens (words, subwords) to vectors
- **Positional Encodings:** Adds order information to tokens

$$
\begin{aligned}
PE_{\text{pos},2i} &= \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right), \\
PE_{\text{pos},2i+1} &= \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)
\end{aligned} \tag{15}
$$

Layers

- **Encoder:**
  - Layer Norm, Multi-Head Attention, Feedforward Network.
  - Produces sequence representations for input.
- **Decoder:**
  - Masked Self-Attention, Cross-Attention, Feedforward Network.
  - Generates output sequence step by step.

The architecture is demonstrated in Figure 7.

1.6. Unsupervised Approaches

1.6.1. Variational Auto Encoder

Generative model to identify underlying distribution

Learns distribution parameters with sampled errors through an encoder decoder framework

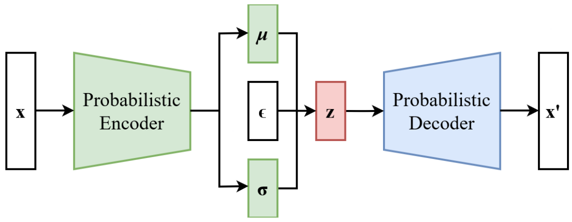


Figure 2. Variational Auto Encoder [6]

- Generate lower dimensional embeddings
- Remove biases from data
- Identify outliers by checking encodings

2. Natural Language Processing

2.1. Terminology

N-grams: Phrases of n tokens

Dictionary: Set of all words

Document: Single sample of text

Corpus: Set of all documents

Stop Words: commonly used filler words (removed)

2.2. Preprocessing

Common steps to prepare text

Tokenization: Split into words, sub words

Stemming: Short form (improving -> improv)

Lemmatization: Root word (improving -> improve)

2.3. Embedding

2.3.1. Vectorization

Capture occurrences and frequency of appearances

Bag of Words: Frequency counts of words

One Hot Encoding: Binary Vectors for Tokens

TF-IDF: Term Frequency · Inverse Document Frequency

TF-IDF

$$
\begin{aligned}
\text{TF}(t, d) &= \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \\
\text{IDF}(t) &= \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right) \\
\text{TF-IDF}(t, d) &= \text{TF}(t, d) \cdot \text{IDF}(t)
\end{aligned} \tag{16}
$$

2.3.2. Word Level

Capture semantic representations

Word2Vec: ML Model for Embeddings

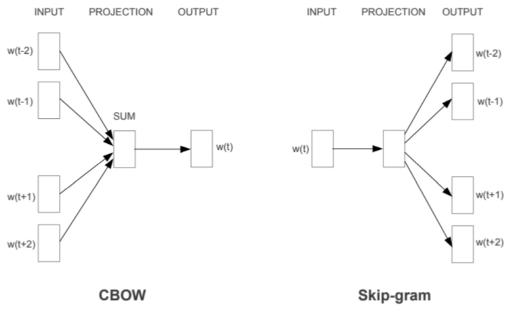


Figure 3. Word2Vec Models
[7]

Table 4. Word2Vec

	CBOW	**Skip Gram**
Input	Context	Token
Output	Token	Context
Embeddings	Input Layer	Output Later
Pros	Faster	Smaller Datasets
Cons	Rare words	Slower

Negative Sampling: Add a few unrelated target words to reduce overall updates. Model better to distinguish.

GLoVe: Global Vectors for Word Representation.
Factorization of co-occurence matrix

- Preserves global corpus
- Computationally efficient

2.3.3. Sentence Level

BERT: Bidirectional Encoder Representations from Transformers
Relies on Bidirectional Attention

- Masked Language Modelling (MLM)
  Randomly mask tokens and predict probability

$$\mathcal{L}_{\text{MLM}} = -\sum_{i \in M} \log P(x_i | \hat{x}) \tag{17}$$

- Next Sentence Prediction (NSP)
  Predict if a sentence follows another

$$\mathcal{L}_{\text{NSP}} = -\sum_{i} y_i \log P(y_i) + (1 - y_i) \log(1 - P(y_i)) \tag{18}$$

[CLS] token at the beginning of the sentence is a pooled representation.

Sentence BERT

- Generates Fixed Size Sentence embeddings
- Relies on Siamese Networks

Siamese Networks

- Two Sentences passed through the same BERT model
- Similarity over pooled sentence embeddings

Contrastive Loss

- Pairs of Sentences
- Calculate distance (d) between embeddings
- Classify labels (y) [1 Similar, -1 dissimilar]

$$\mathcal{L}_{\text{contrastive}} = \frac{1}{2}\left(y \cdot d^2 + (1 - y) \cdot \max(0, \text{margin} - d)^2\right) \tag{19}$$

Triplet Loss

- Anchor (u), with positive (v^+) and negative (v^-) sentences
- Similarity scores (Cosine distance)

$$\mathcal{L}_{\text{triplet}} = \max(0, \text{sim}(u, v^-) - \text{sim}(u, v^+) + \text{margin}) \tag{20}$$

Margin: Hyper-parameter for separation

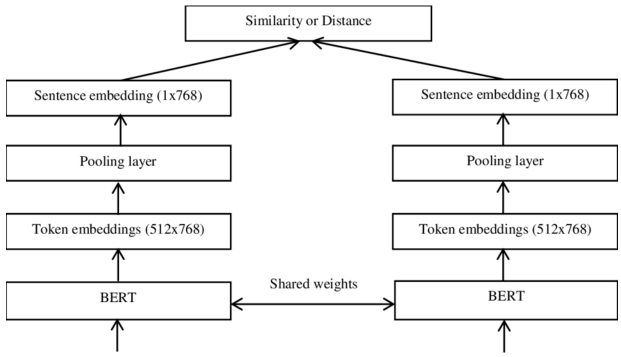


Figure 4. SBERT Siamese Network
[4]

2.4. Feature Retrieval

2.4.1. Named Entity Recognition

Labelling entities to predefined categories

Input: Barack Obama was born in Hawaii.
Output: B-PER I-PER O O O B-LOC

- Calculate word embeddings (Hashed Bloom embeddings)
- Apply context (LSTM or iterative CNN)
- Pass through Attention based classifier (Context words and entities as features)

Bloom Embeddings: Multiple hashes
Custom Entities: Calculate embeddings and check for similarity scores across all entities

2.4.2. POS Tagging

Identifying the grammatical role of words. Trained similar to NER

Input: The quick brown fox jumps over the lazy dog.
Output: DT JJ JJ NN VBZ IN DT JJ NN

2.5. Other Models

2.5.1. Natural Language Inference

Also called Recognizing Textual Entailment (RTE)
Determine relationship between Premise and Hypothesis.
Predict whether hypothesis is Entailed, Contradicted or Neutral.

2.5.2. Topic Modelling

Unsupervised approach to identify topics in text

Latent Dirichlet Analysis (LDA)
Generative probabilistic model

- Specify number of topics
- Randomly assign topic to each word in each document
- Assign topic to document based on word assignments
- Update document topic probabilities sampling from topic contributions
- Iterate until convergence

$$p(z_{d,n} = t | w_{d,n}, d) \propto \frac{N_{t,d} \cdot N_{w,t}}{N_t} \tag{21}$$

where:

- $N_{t,d}$ is the number of words assigned to topic t in document d,
- $N_{w,t}$ is the number of times word $w_{d,n}$ is assigned to topic t,
- N_t is the total number of words assigned to topic t.

Latent Semantic Index (LSI)
Identify hidden topic relations with Singular Value Decomposition

- Select number of topics (k)
- Build Document Term Frequency matrix
- Perform Singular Value Decomposition
- Extract topic and document vectors for comparison

$$A \approx U\Sigma V^T \tag{22}$$

where:

- $U \in \mathbb{R}^{m\times k}$ is the term-topic matrix,
- $\Sigma \in \mathbb{R}^{k\times k}$ is the diagonal matrix of singular values,
- $V^T \in \mathbb{R}^{k\times n}$ is the document-topic matrix.

BERTopic
Transformer-based approach

- Document embeddings (BERT)
- Dimensionality Reduction (UMAP)
- Clustering (HDBScan)
- Topic Representations (Words with highest TF-IDF scores in each cluster)
- Aggregate cluster for each word in document for topics

Supervised Approach

- Add labels as documents to embed
- Cluster with labels as centroids

3. Time Series

3.1. Analyses

3.1.1. Decomposition

Time Series Decomposition of Components

$$\begin{aligned} \text{Additive: } Y_t &= T_t + C_t + S_t + R_t \\ \text{Multiplicative: } Y_t &= T_t \cdot C_t \cdot S_t \cdot R_t \\ \text{Multiplicative: } \log Y_t &= \log T_t + \log C_t + \log S_t + \log R_t \end{aligned} \tag{23}$$

T_t: Trend
Long-term movement of data. Calculated by smoothing series

- Moving Averages
- Locally Estimated Scatterplot Smoothing (LOESS) - Regression

S_t: Seasonality
Regular fixed-term patterns

- De-Trend the series
- Group by Time Period (January, Monday)
- Calculate average of group
- Center around 0 (Subtract overall mean)

C_t: Cyclic
Long-term oscillations not fixed in period

- Remove trend and seasonality
- Smooth the residual data
- Fourier Transform to identify dominant peaks
- Reconstruct Cyclic with dominant peaks

R_t: Residuals
Noise. Remainder component.

3.1.2. Stationarity

Statistics (mean, variance, covariance) don't change over time

- Strict: Probability Distribution constant
- Weak: Statisitic measures remain constant (Common)

Augmented Dickey-Fuller Test (ADF)
H_0: $\gamma = 0$ [Unit Root] H_1: $\gamma < 0$

$$\Delta Y_t = \alpha + \beta t + \gamma Y_{t-1} + \sum_{i=1}^{p} \delta_i \Delta Y_{t-i} + \varepsilon_t \tag{24}$$

Differencing
Differencing a time series to stationarize

$$\begin{aligned} \Delta Y_t &= Y_t - Y_{t-1} \text{ (or)} \\ \Delta Y_t &= \frac{Y_t - Y_{t-1}}{Y_{t-1}} \text{ (Log difference)} \end{aligned} \tag{25}$$

3.1.3. Autocorrelation

Understand correlation structure

Auto-correlation Plot (ACF)
Correlation of value Y_t with all values $Y_{t-1} \ldots Y_{t-k}$

$$\begin{aligned} \rho_k &= \frac{\sum_{t=1}^{n-k}(Y_t - \bar{Y})(Y_{t+k} - \bar{Y})}{\sum_{t=1}^{n}(Y_t - \bar{Y})^2} \\ \text{CI} &= \pm \frac{t_{\alpha/2}}{\sqrt{n}} \end{aligned} \tag{26}$$

ACF confidence interval increases over lags since fewer observations are available as we look further back.

Partial Auto-correlation Plot (PACF)
Correlation of value Y_t with value Y_{t-k} without intermediary lags
Fits an OLS model

$$\begin{aligned} Y_t &= \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_{k-1} Y_{t-(k-1)} + \epsilon_t \\ \phi_k &= \hat{\beta}_k \\ \text{CI} &= \pm \frac{t_{\alpha/2}}{\sqrt{n}} \end{aligned} \tag{27}$$

3.2. Forecasting

Fitting time series models and forecasting future values

3.2.1. Exponential Smoothing

Time series with Trend and Seasonality

$$\hat{y}_{t+1} = \alpha y_t + (1-\alpha)\hat{y}_t \tag{28}$$

Double Exponential (Trend)

$$\begin{aligned} \hat{y}_{t+1} &= L_t + T_t \\ L_t &= \alpha y_t + (1-\alpha)(L_{t-1} + T_{t-1}) \\ T_t &= \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \end{aligned} \tag{29}$$

Triple Exponential Holt Winters (Seasonality)

$$\begin{aligned} \hat{y}_{t+1} &= (L_t + T_t) \cdot S_{t+m} \\ L_t &= \alpha \frac{y_t}{S_{t-p}} + (1-\alpha)(L_{t-1} + T_{t-1}) \\ T_t &= \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \\ S_t &= \gamma \frac{y_t}{L_t} + (1-\gamma)S_{t-p} \end{aligned} \tag{30}$$

Optimization occurs by Gradient Descent

3.2.2. ARIMA

Auto-Regressive Integrated Moving Average
Fits with MLE estimation or Regression

- Use PACF to get Auto-regressive term (p)
- Apply differencing (d) to get stationary data
- Use ACF to get Moving Average term (q)

$$\begin{aligned}
\Phi(B)(1-B)^d Y_t &= \Theta(B)\epsilon_t \\
\text{ARIMA (1,1,1): } Y_t &= \phi_1 Y_{t-1} + (Y_t - Y_{t-1}) + \theta_1\epsilon_{t-1} + \epsilon_t \\
\text{AR: } Y_t &= \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon_t \\
\text{MA: } Y_t &= \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q}
\end{aligned} \tag{31}$$

Residuals should resemble white noise
Assess performance across different ARIMA variants (p, d, q) comparing metrics
Akaike Information Criteria (AIC)

$$\text{AIC} = -2\ln(L) + 2k \tag{32}$$

Bayesian Information Criteria (BIC)

$$\text{BIC} = -2\ln(L) + k\ln(n) \tag{33}$$

- L: Maximum likelihood of the model.
- k: Number of estimated parameters in the model.
- n: Total number of observations in the dataset.

Choose model with a lower AIC / BIC score
SARIMAX:
Seasonal ARIMA with Exogenous Variables

$$\Phi_p(B)\Phi_P(B^m)(1-B)^d(1-B^m)^D Y_t = \Theta_q(B)\Theta_Q(B^m)\epsilon_t + \beta X_t \tag{34}$$

- Seasonality (m): Decompose / Domain knowledge
- Get seasonal p, q from periodic spikes in PACF, ACF
- Seasonal difference subtracts from $t-m$ observation

Exogenous Variables: Dependent Variables (predictors). Required for future time steps for forecast.

Table 5. SARIMAX Components

Component	Variable	Identification
Auto-Regressive	p	Partial Autocorrelation (PACF)
Differencing	d	Augmented Dickey Fuller (ADF)
Moving Average	q	Autocorrelation (ACF)
Seasonality	s	Seasonal Decomposition

3.2.3. GARCH

Generalized Auto-Regressive for Conditional Heteroskedasticity
Models Variance of Time Series

- Fit an ARMA model to the time series
- Fit variance of time series on residuals from ARMA

$$\begin{aligned}
Y_t &= \mu + \epsilon_t \\
\sigma_t^2 &= \omega + \sum_{i=1}^{q}\alpha_i\epsilon_{t-i}^2 + \sum_{j=1}^{p}\beta_j\sigma_{t-j}^2
\end{aligned} \tag{35}$$

Forecasts: Expectation of Errors Variance

$$\begin{aligned}
\sigma_{T+h}^2 &= \omega + \sum_{i=1}^{q}\alpha_i\mathbb{E}[\epsilon_{T+h-i}^2] + \sum_{j=1}^{p}\beta_j\sigma_{T+h-j}^2 \\
\mathbb{E}[\epsilon_{T+h-i}^2] &= \sigma_{T+h-i}^2
\end{aligned} \tag{36}$$

Selection of (p, q)

- Fit an initial GARCH (1,1)
- Use ACF, PACF of squared residuals

Evaluation
Squared returns act as a good proxy when variance is not available. Can be compared visually / with correlation.
NOTE: Conventional ML / RNNs can also be used to forecast stepwise, but lack the inbuilt capabilities to capture more temporal components

References

[1] *Activation Functions*. [Online]. Available: https://medium.com/the-modern-scientist/an-overview-of-activation-functions-in-deep-learning-97a85ac00460.

[2] *Neural Networks*. [Online]. Available: https://www.cs.toronto.edu/~lczhang/360/lec/w02/terms.html.

[3] *Recurrent Neural Networks*. [Online]. Available: https://www.linkedin.com/pulse/rnn-lstm-gru-why-do-we-need-them-suvankar-maity-joegc/.

[4] *Siamese*. [Online]. Available: https://www.researchgate.net/figure/SBERT-Siamese-model-architecture_fig3_378440592.

[5] *Transformer Architecture*. [Online]. Available: https://quantdare.com/transformers-is-attention-all-we-need-in-finance-part-i/.

[6] *VAE*. [Online]. Available: https://towardsdatascience.com/uncovering-anomalies-with-variational-autoencoders-vae-a-deep-dive-into-the-world-of-1b2bce47e2e9.

[7] *Word2Vec*. [Online]. Available: https://towardsdatascience.com/nlp-101-word2vec-skip-gram-and-cbow-93512ee24314.

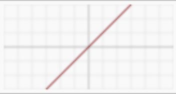
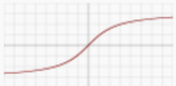
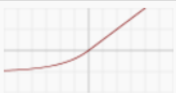
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Figure 5. Activation Functions
[1]

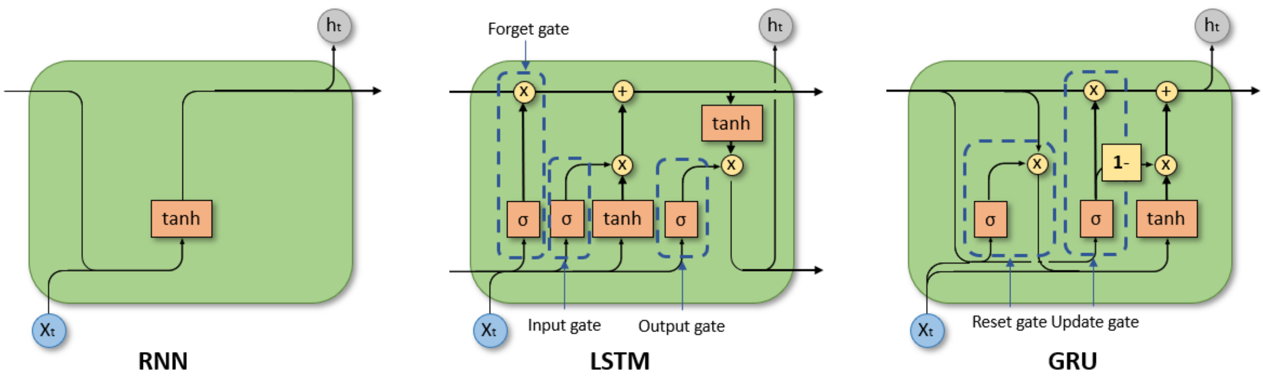


Figure 6. Recurrent Neural Networks
[3]

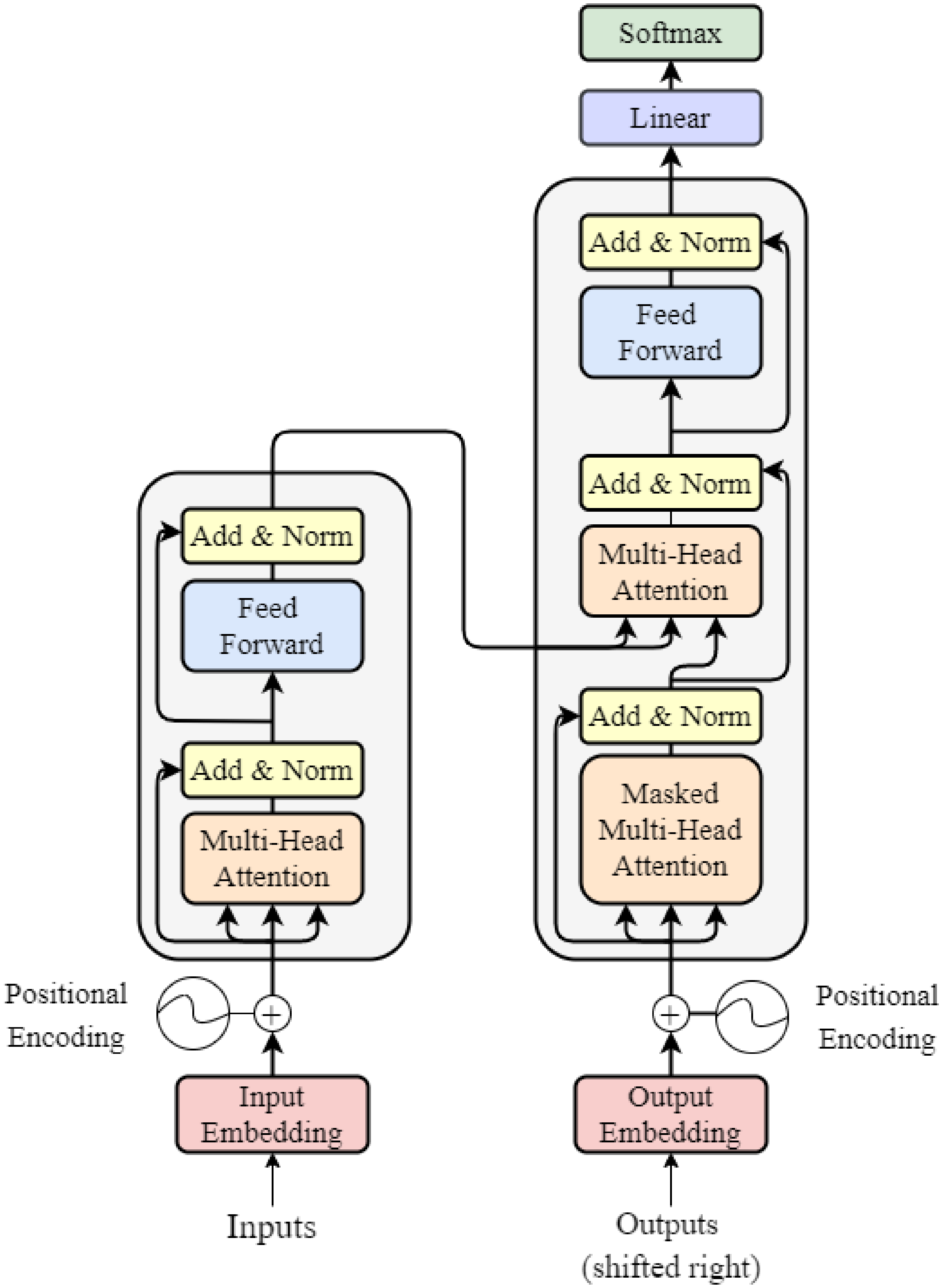


Figure 7. Attention Based Transformer
[5]