Title - *'ASL Using Transformer Model'*

Github Link - Repository

Dataset - The data consist of ~200 gigabytes of parquet files which consists of sequence points represented along a 3D space (x,y,z). The data set belongs to Google ASL Dataset.

The Dataset was generated/collected from the deaf and dumb community where a camera was placed to capture what they are trying to communicate using American Sign Language. There are 1630 feature columns originally in one parquet file. Rows may vary because of phrases of length.

Parquet - Parquet is an open source file format built to handle flat columnar storage data formats. Parquet operates well with complex data in large volumes.It is known for its both performant data compression and its ability to handle a wide variety of encoding types.

Working with parquet files can be a little challenging. But pandas and tensorflow offer specific techniques. Working with parquet files can be a little challenging. But pandas and tensorflow offer specific methods.

Data Pre-Processing - The data provided to us is in .parquet format. The size of this file is too big for computation. We have converted this .parquet file into a .tfrecord file to process the data. The number of files in both extensions will be the same.

After converting the data from .parquet to .tfrecord the training data was reduced to roughly around 5 gigabytes. The data was further split into 80:20 to train the model and to validate it. Few of the values were Nan, that was dealt by updating it to zero at that particular row and column of the cell.

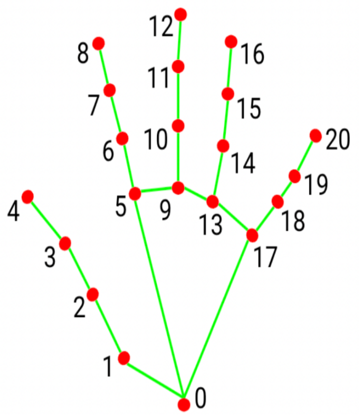
To deal with the phrases of uneven length we have used padding.

Feature Extraction - As told earlier the data is present along the x,y,z axis. There are 1630 columns of features. Few of the columns are not related to our needs. For example, we don't need the sequence points related to legs or the face. They are ir-relevant for the ASL model. The data we need to train our model resides along the palm and the hand.

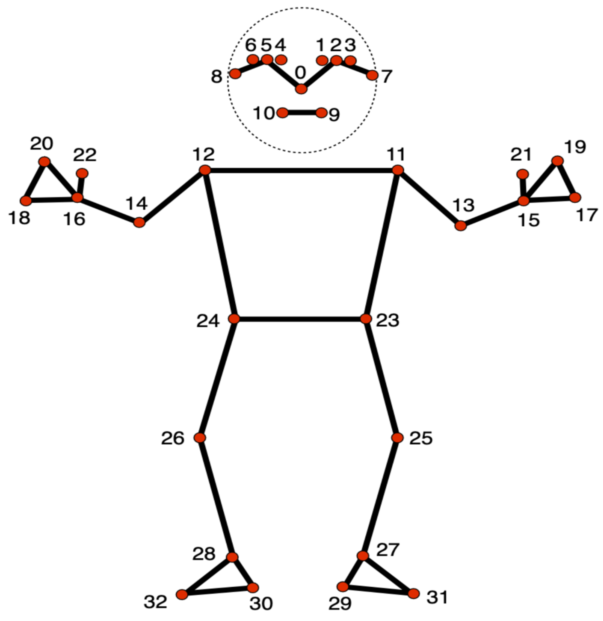
We are going to extract the columns from the hand. There are various landmarks present inside the file. We need to pick the landmarks which are titled such as x_right_hand_0, y_right_hand_0, z_right_hand_0. These are just sample landmarks which we want to extract. If we count the total feature columns we will get from this extraction process it is 156 feature columns which hold the data regarding the sequence points in two different hands.

X and Y axis the data is normalised between 0.0 to 1.0 with respect to image width and height. The Z axis represents the distance between the camera and the hand.

Mediapipe api is being used to capture these sequence points and landmarks. Below is the pictorial representation of the hand and human body with respective sequence numbers provided by the api.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP



Data Visualization - The dataset consist of data from alphabet a-z and numbers from 0-9 and few symbols like @,#,$ etc. There are around 59 different characters present inside the phrase.

There are a few phrases which were repeated by various participants such as ‘why do you ask silly questions’ was repeated 117 times in the data set. The average length of words spoken by a person in the data is 25 words.

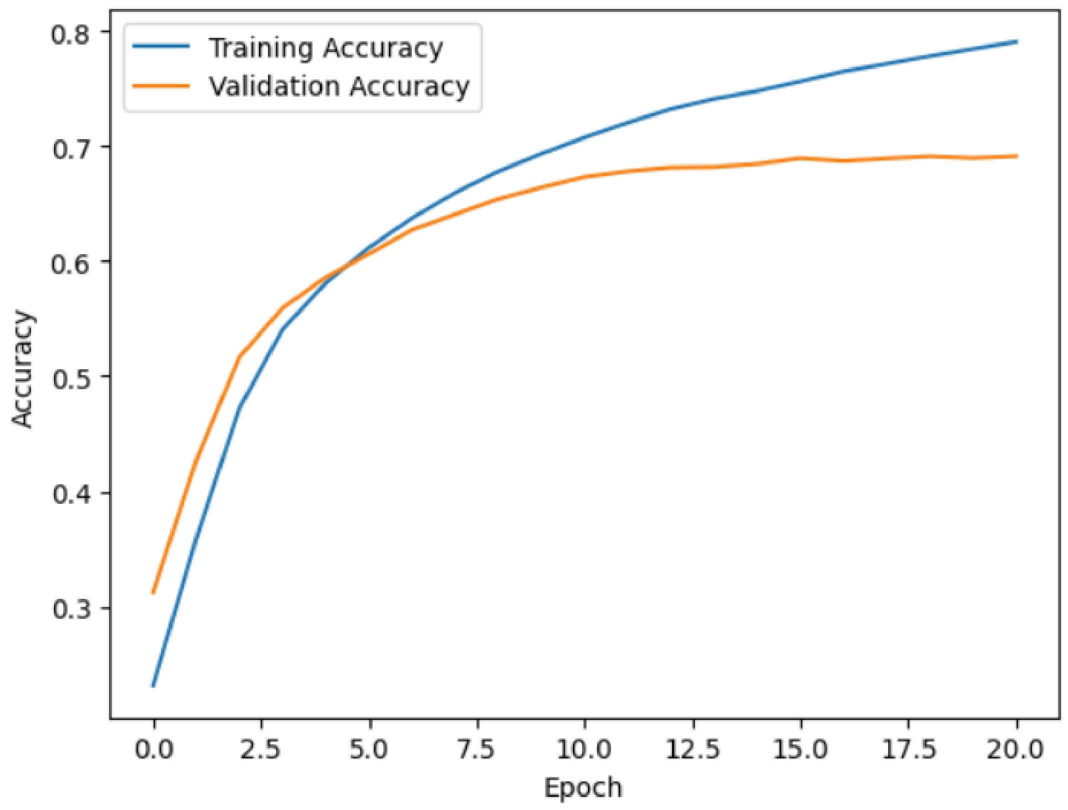
Train & Validation-The code sets up training and validation datasets for a machine learning model using TensorFlow. Both datasets are derived from TFRecord files. The training dataset comprises 80% of the files, with a batch size of 64. Similarly, the validation dataset is created from the remaining 20% of files, also with a batch size of 64. Both datasets undergo decoding and conversion processes. The datasets are then batched, prefetched for efficiency, and cached in memory for faster access during training.

Transformer model Overview-
We have try design Transformer model from scratch without using any pretrained model following are the some component of our design
a.)TokenEmbedding & LandmarkEmbedding-These layers are likely part of the feature extraction stage in a transformer model designed for ASL fingerspelling recognition. The `TokenEmbedding` layer captures token-level information, and the `LandmarkEmbedding` layer extracts hierarchical features from the input, contributing to the model's ability to understand and recognize ASL fingerspelling sequences.

b.)Transformer Encoder Layer-`TransformerEncoder` layer plays a central role in transforming raw ASL fingerspelling data into a rich and meaningful representation, facilitating subsequent stages of the model for accurate recognition. We use Self-Attention Mechanism in our encoder layer

c.)Decoder Layer-This customised TransformerDecoder layer enhances ASL fingerspelling recognition, integrating self-attention, layer normalisation, and feed-forward mechanisms. Causal attention masks prevent future information flow, contributing to context-aware decoding. Dropout and layer normalisation aid training stability and generalisation.

d.)Training of Transformer Model-
This code trains a Transformer model for ASL fingerspelling recognition. The model has 200 hidden units, 4 attention heads, and a feed-forward dimension of 400. It processes input sequences of length `FRAME_LEN` and generates output sequences of maximum length 64. The encoder consists of 2 layers, the decoder has 1 layer, and the model outputs 62 classes. Categorical cross entropy loss with label smoothing (0.1), Adam optimizer (learning rate 0.0001), and early stopping (patience 5) are employed for training. Training occurs over 40 epochs.
After this we are getting validation accuracy nearly about 68.96%



The above graph indicates a potential issue with the model's ability to generalise beyond the training data, suggesting the need for further exploration into hyperparameter tuning, regularisation techniques, or potential data-related challenges to improve overall model performance on unseen data.

e.)Hyperparameter tuning-

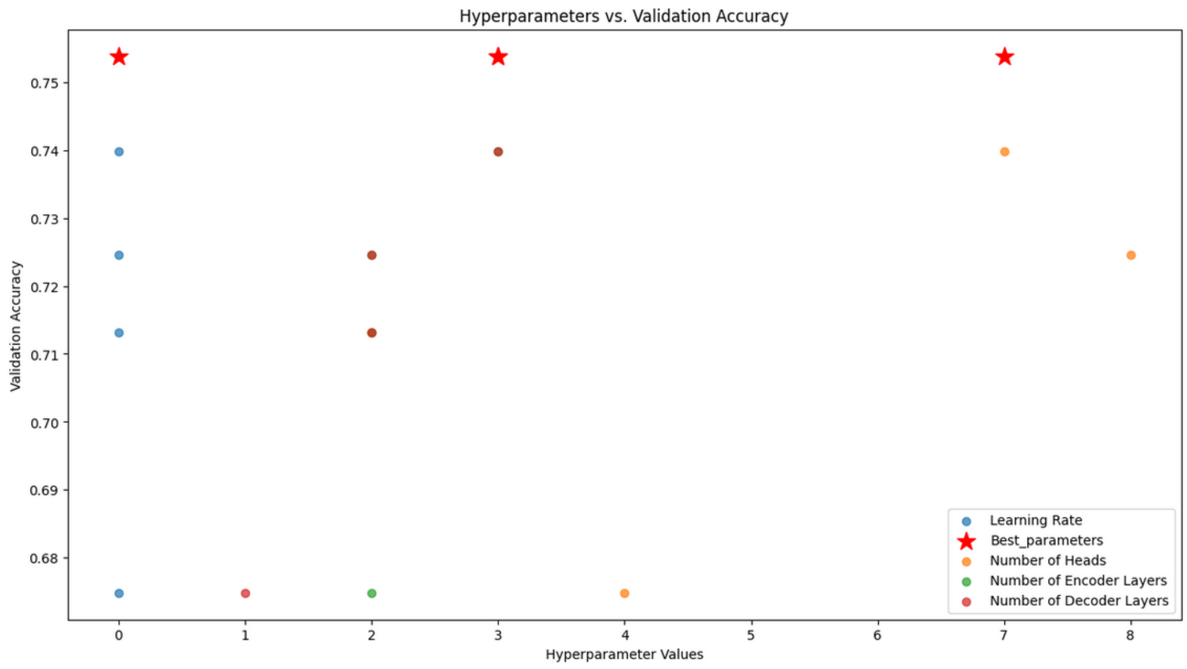
As we build a transformer model with random parameters from scratch therefore our first and obvious work is to vary the parameters and find the best parameter out of it . Although due to lack of computation power we are able to vary the minimum parameter range .

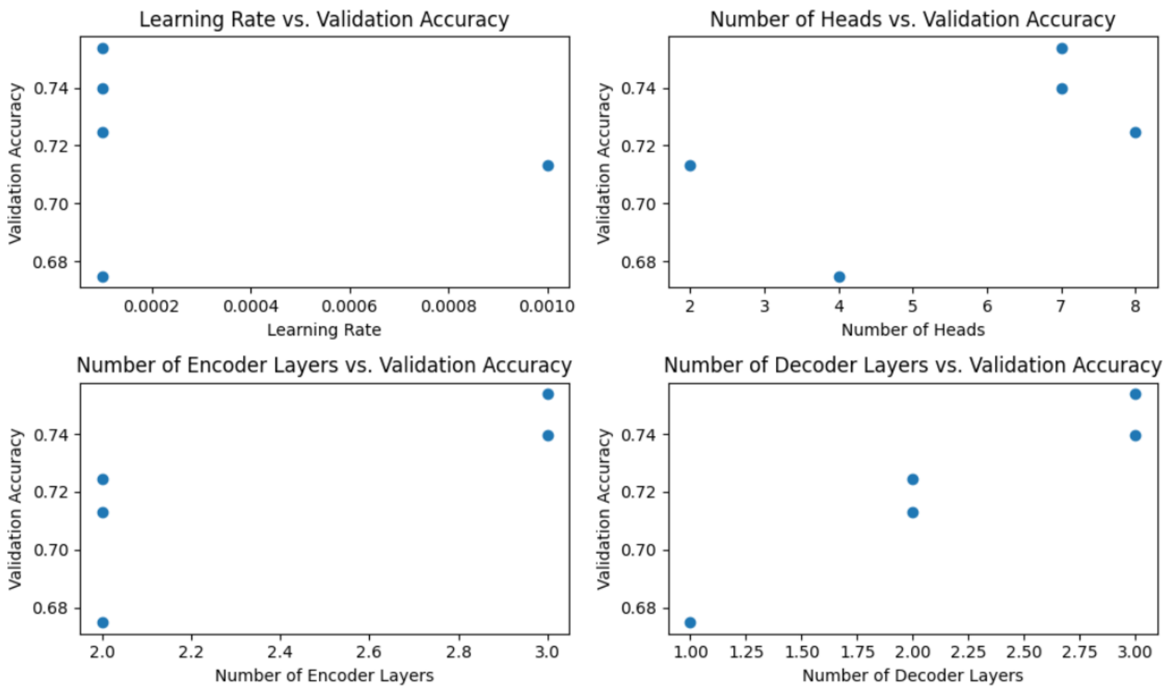
Parameters:

- Hidden Units (`num_hid`): Ranges from 100 to 400 in steps of 50.
- Attention Heads (`num_head`): Ranges from 2 to 8 in steps of 1.
- Feed-Forward Dimensions (`num_feed_forward`): Ranges from 200 to 800 in steps of 100.
- Encoder Layers (`num_layers_enc`): Varies from 1 to 3 in steps of 1.
- Decoder Layers (`num_layers_dec`): Varies from 1 to 3 in steps of 1.
- Learning Rate (`learning_rate`): Chooses from 1e-4, 1e-5, and 1e-6.

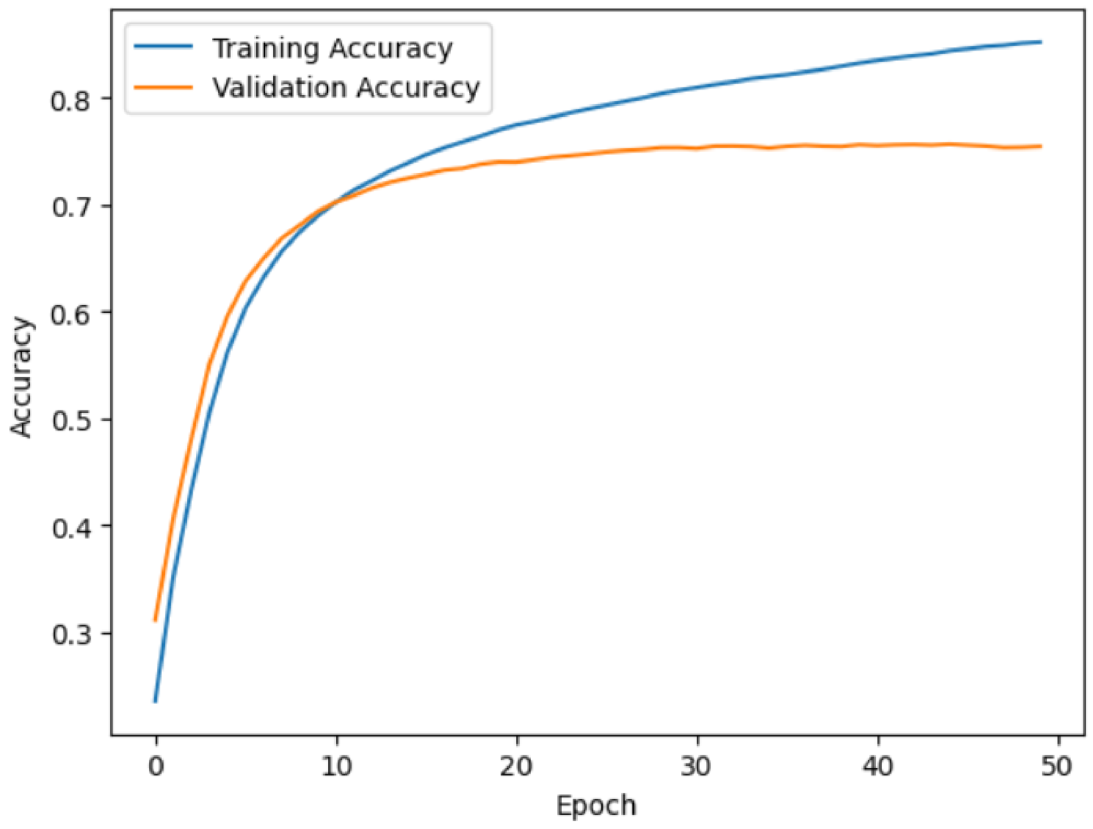
The tuning process runs for 30 epochs, with early stopping based on validation accuracy. The best hyperparameters are extracted and used to build, compile, and train the final model over 100 epochs.

Best parameters and accuracy plot something look like this





After that we were able to improve our validation accuracy to 75% .



Scope for improvement-

- The model is predicting small phrases much more accurately than the long phrases.
- To solve the above problem we can create a box with 1-6 length sentences and a box with 7-12 length sentences. In that way we can reduce our error to detect long phrases accurately.
- Despite having the left-right hand separately we can overlap both the hands in a single feature column so that we don’t need to pick the dominant hand separately.
- We can use dynamic FRAME Length instead of a fixed FRAME Length