Sign Language Translation

Pranav Balaji
CSIS Department
BITS Pilani, Hyderabad Campus
Hyderabad, India
f20190040@hyderabad.bits-pilani.ac.in

Vraj Gandhi
CSIS Department
BITS Pilani Hyderabad Campus
Hyderabad, India
f20190158@hyderabad.bits-pilani.ac.in

Abhinav Rahul
EEE Department
BITS Pilani, Hyderabad Campus
Hyderabad, India
f20191354@hyderabad.bits-pilani.ac.in

Fenil Bardoliya
CSIS Department
BITS Pilani, Hyderabad Campus
Hyderabad, India
f20190152@hyderabad.bits-pilani.ac.in

Kenute Joseph
CSIS Department
BITS Pilani, Hyderabad Campus
Hyderabad, India
f20190119@hyderabad.bits-pilani.ac.in

Suchit Uppala
CSIS Department
BITS Pilani, Hyderabad Campus
Hyderabad, India
f20190115@hyderabad.bits-pilani.ac.in

***Abstract*—The hearing and speech impaired community doesn't have much technology that connects them with the rest of the world because the concept of Sign Language detection is overlooked. Machine Learning and Image classifiers can be used via which the computer can recognize sign languages, and can then be interpreted by people. Convolutional Neural Networks (CNNs) can be used to process static images, and sequence2sequence models can be used to process videos. In this paper we fine-tune the pre-trained Inception v3 on a dataset consisting of static images with sign alphabets captured by an RGB camera. We later extend the problem by translating video sequences from the Phoenix14T dataset using a transformer encoder-decoder architecture.**

***Index Terms*—Sign Language Translation, Image processing, Video processing, Inception V3, T5**

I. Image translation

A simpler problem to tackle before moving on the main task of video translation is image translation. The first part of this paper, static sign recognition can be termed as a classification task. Here, the input is an image consisting of a hand signing an alphabet, and the output is a probability distribution over the 29 classes (26 alphabets + del + space + nothing). We implemented [1] for this problem.

II. Dataset and preprocessing

This kaggle dataset was used instead of the dataset used in [1] as it had 3000 images for each alphabet whereas the latter had only 63 for each alphabet. It consists of static sign language gestures (letters) captured on an RGB camera. The Inception v3 model cannot process images smaller than 299x299. Since the dataset consisted of 200x200 size images, we resized them to the appropriate dimensions using linear interpolation. Following industry standards, we then split the images into training , testing and validation sets

III. CNN architecture

The reason we used CNNs for this is that in a normal neural network, when we feed an image as an input, each pixel in the image will be connected to a neuron in the input layer. This leads to a extremely large number of neurons and subsequently a large number of weights to be updated. In case of CNNs, the layers are organized in 3 dimensions: height, width and depth. The neurons in one layer do not connect to all the neurons in the next layer but only to a small region of it. The final output will be reduced to a single vector of probability scores, organized along the depth dimension. In any image, there are a lot of pixels that carry unwanted information. CNNs work around this by performing a series of operations such as convolution and pooling to repeatedly reduce the dimension of a patch of image by using filters. Another advantage of CNNs is the translational and rotational invariance they provide.

IV. Inception V3 architecture

For predicting Image2Text, we use the Inception-v3 [2] model which is a pre-trained CNN model used for image recognition. This model has 48 layers and has shown an accuracy greater than 78.1 on the ImageNet dataset. We use the Inception-v3 pre-trained model and train it on our dataset.

V. Experimental setup

A. Loss

The standard Cross entropy loss is used as is common in classification problems.

B. Training

The pre-trained Inception v3 model was used, but the final linear layer was replaced to a new layer that maps from the previous output dimension to num_classes (In this case 29). The Adam optimizer with a learning rate of 0.001, β_1 = 0.9, β_2 = 0.999, ϵ = 10^{-8} was used with no weight decay. The model was trained on the entire training dataset with a batch size of 32 for 2 epochs.

VI. Results and discussion

Dataset	Accuracy
train	98.83%
val	98.74%
test	97.59%

As shown in the table, we have achieved an accuracy of 97.58% on the test set, which is almost 5% greater than the paper we implemented. This can mostly be attributed to a larger dataset and more training time.

VII. Video Translation

Sign language translation is a non-trivial task of interpreting text sequences from video sequences. Like other translation problems, the input word order is usually not the same as the output word order. Deaf people cannot communicate by using the signs of alphabets. Instead they have a complex set of gestures to convey sentences with proper grammar. To solve that problem, we will now translate the videos containing sign language gestures into their corresponding sentences. So as a step towards solving the true real world problem we seek to process these gestures and not just single images. So now our problem statement is changed from translating images to translating videos and predicting proper syntax of the translated words to create a coherent sentence. Both the input and the output space of the new problem are different. This is no longer a classification problem but rather a Seq2Seq problem.

[3] addresses this problem using RNNs. An improvement over this would be using transformers instead of LSTMs and that is what we did.

The new model will have an encoder-decoder transformer with attention mechanism architecture which takes output of the CNN as its input. This model will predict the gloss and text of the video and compare it to the ground truth.

VIII. Dataset and preprocessing

We used the PHOENIX14T dataset introduced in [3] for training and evaluating the model. It consists of 7046 videos for training, 519 videos for validation and 642 videos for testing. Each instance of example consists of three fields:

1) CNN embeddings of each frame in the video as a PyTorch tensor
2) Ground truth Glosses for the video
3) Ground truth Translation

Glosses are a form of human-readable intermediate representation of the sign videos. A gloss consists of a sequence of words in the order that they were signed, without consideration for grammar. In contrast, the translation is a natural language sentence. The sign videos are in German Sign Language and the Glosses and translation are in German.

IX. Architecture

We use an attention based transformer encoder-decoder architecture first proposed in [4] shown in fig 1 which to accept the last layer of the CNN for the embeddings of the video frames which is trained to predict the German translation of the signs in the frames of those videos.

The end-to-end model consists of 3 parts:

1) CNN: Takes in raw pixel data of frames and generates embeddings.
2) Transformer encoder layer: Takes the embeddings produced by the previous layer and assign attention to these embeddings before passing it on to the decoder layer.
3) Transformer decoder layer: Takes the attended embeddings and autoregressively generates the final text translation word-by-word.

Apart from these main layers, there is another utility linear layer which transforms the transformer encoder output to a probability distribution over gloss vocabulary, which is explained in the next section

X. Loss function

Inspired from [3], we use a two level loss.

1) Gloss loss (L_g): Inspired from [3], the encoder outputs are passed through a linear layer which maps each frame embedding to a probability distribution over the entire gloss vocabulary. This enables the model to predict a gloss per frame. Since the number of frames are numerous compared to the ground truth glosses, CTC Loss is used to calculate this loss.
2) Translation loss (L_t): The decoder has an inbuilt linear layer which maps from the decoded embeddings to the german vocabulary. These predicted terms are passed through a atndard Cross Entropy loss function against the ground truth translation as is common practise in Natural Language Processing reseach.

The two losses are weighed relatively and added together to produce the final loss with which to perform backpropagation.

$$L = \lambda_g L_g + (1 - \lambda_g) L_t$$

Where λ_g is a hyperparameter to by tuned.

XI. Experimental Setup

The following encoder-decoder architectures were used for building the model:

1) BERT [5] pre-trained german-uncased encoder with our own implementation of a decoder
2) Our own implementation of both encoder and decoder
3) Pre-trained T5-small [6] with encoder-decoder layers.

A. Tokenization

Any model that deals with text needs to have a tokenizer that maps words or parts of words to numbers The dataset already has the trained CNN layers of the images, so we needed to pass these frame embeddings through the encoder and then the subsequent output through the decoder.

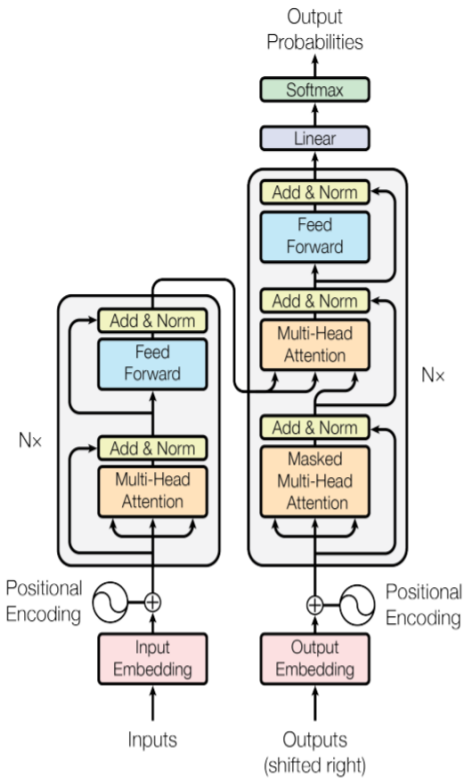

Fig. 1. Transformer architecture

B. Configurations

We trained and tested the model using the following 5 configurations:

1) Configuration 1:
   a) λ_g = 0.01
   b) dropout rate = 0.1
2) Configuration 2:
   a) λ_g = 0.01
   b) dropout rate = 0.25
3) Configuration 3:
   a) λ_g = 0.05
   b) dropout rate = 0.1
4) Configuration 4:
   a) λ_g = 0.1
   b) dropout rate = 0.1
5) Configuration 5:
   a) λ_g = 0.5
   b) dropout rate = 0.1

The models' learning rate was set to 10^{-4} and the extra layers' learning rate was set to $5x10^{-3}$. The models were trained for 15 epochs and were tested after every 300 batches of training.

XII. Results and Discussion

We printed the gloss values from the encoder before passing them through the decoder to see if the glosses were coherent or not. The first two models gave only blank values even on training them with the same gloss values 100 times each. This meant that the BLEU-4 score would be near zero as even increasing the learning rate did not alter the output.

The model we created using the t5-small pre-trained model gave coherent glosses instead of just blanks. So we proceeded to train the model using the 5 configurations mentioned above.

Fig 2 shows the BLEU-4 scores for each configuration for the train, validation and test sets. Configurations 1, 3 and 4 show comparable performance while config 2 demonstrates slow learning. Config 5 does not perform very well and this can be attributed to overweighing the gloss loss compared to translation loss. Fig 3 shows the training loss history for config 3. Since config 3 was the most promising, we ran this config for double the number of epochs (30) and tested the bleu scores every epoch. Fig 4 shows the results of this experiment. Although the training performance is going up drastically compared to the validation and testing performance, we have not reached the point of overfitting since they are not worsening.

Dataset	BLEU-1	BLEU-2	BLEU-3	BLEU-4
train	51.06	30.75	20.64	15.29
val	47.50	26.31	17.60	13.19
test	44.48	24.95	16.45	12.75

XIII. Conclusion and Future work

In this paper, we first created a model to predict alphabets from images of the corresponding sign languages and got an accuracy of 92.86%. Next we set our eyes on translating videos containing sign language gestures to sentences using a CNN + (Encoder-Decoder with attention Transformer) architecture. This was an improvement over the CNN+RNN architecture used previously. We tried to implement the model using pre-trained BERT and also our own encoder and decoder but these two models failed to generate coherent glosses. The model that we created using the t5-small pre-trained model gave gloss outputs that were coherent and as such we trained the model separately with 5 configurations and trained the model further for the configuration with the highest BLEU-4 scores. On looking at the total loss graph, we find that even this model does not converge after training for 30 epochs. We strongly believe that using a larger model and training it for longer using a larger dataset will produce must stronger results. We leave this for future implementation as we do not currently have the resources required to test this.

References

[1] A. Das, S. Gawde, K. Suratwala, and D. Kalbande, "Sign language recognition using deep learning on custom processed static gesture images," pp. 1–6, 2018.

[2] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[3] N. C. Camgoz, S. Hadfield, O. Koller, H. Ney, and R. Bowden, "Neural sign language translation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7784–7793, 2018.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[6] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019.

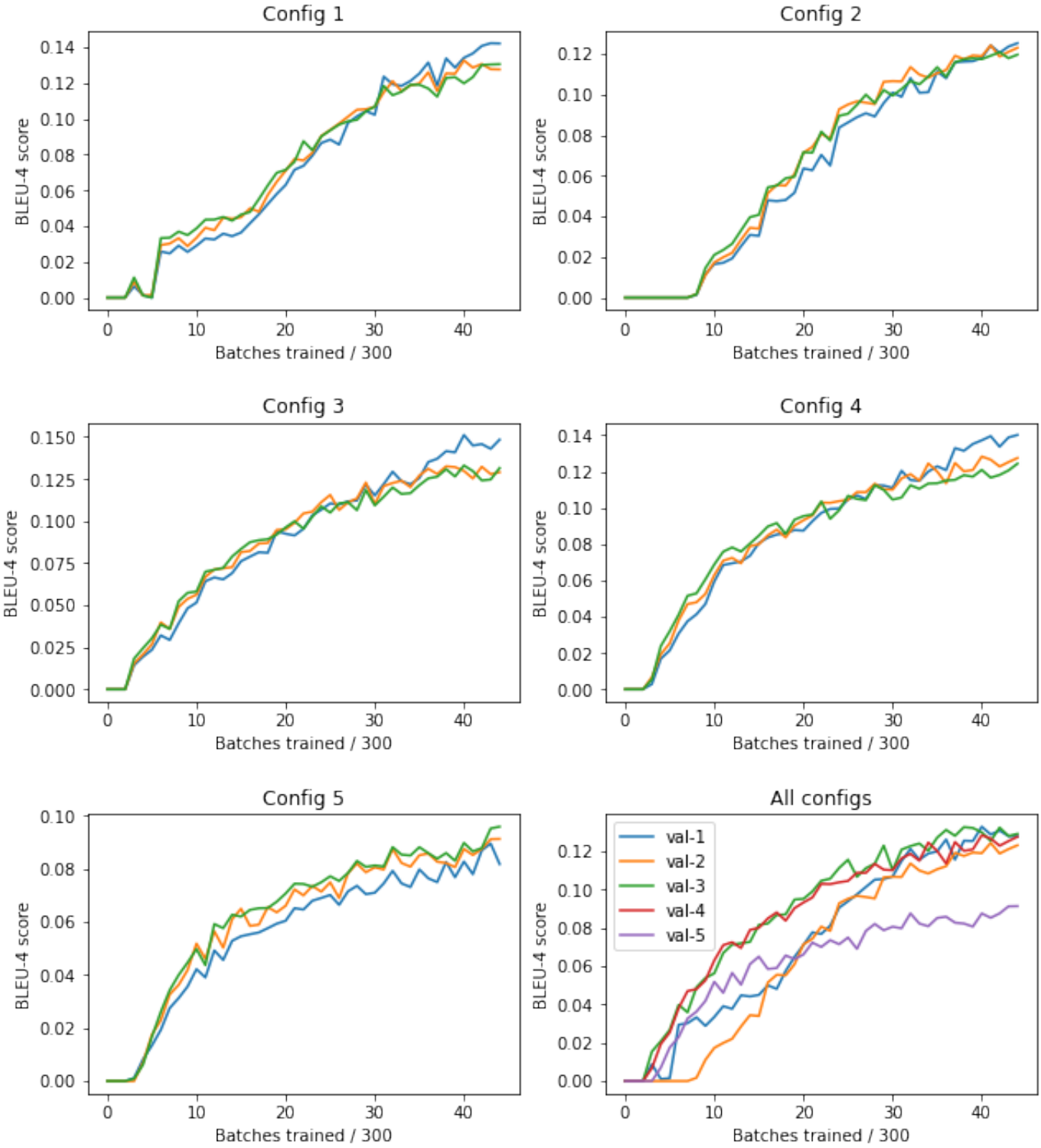

Fig. 2. BLEU history for all configs

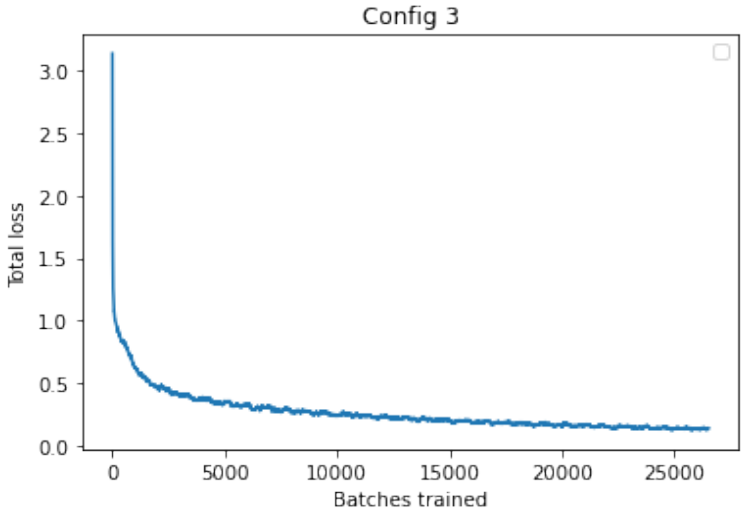

Fig. 3. Training loss history for config 3

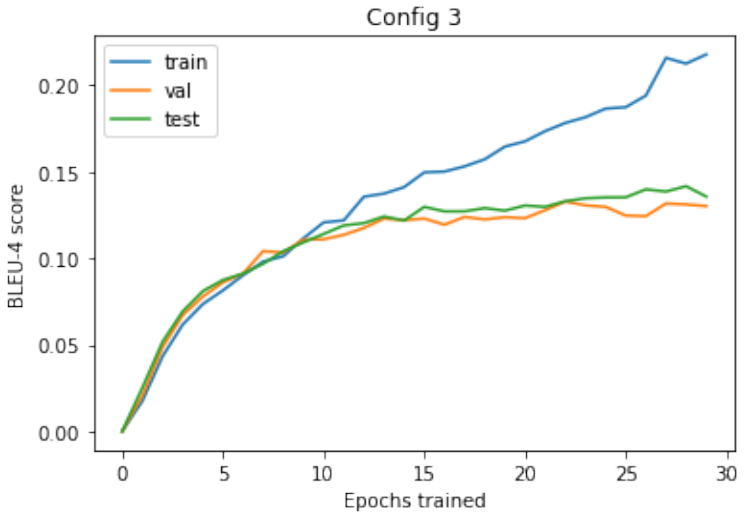

Fig. 4. Training loss history for config 3 ran for 30 epochs

CONTRIBUTIONS

Name	Contribution
Pranav B	Project management, Base implementation, Improvement idea, implementation of new model, testing and training, report writing
Vraj Gandhi	Improvement planning and implementation, report writing
Abhinav Rahul	Base implementation, its testing, report writing
Kenute Joseph	Report writing
Fenil Bardoliya	Report writing
Suchit Uppala	Report writing

Fig. 5. Contributions