AI-Based Gesture Animation for Sign Language Avatars

Flavien ROMANETTI 73100

Summer 2025

This project aims to create sign language gestures animation using machine learning. Two model were created, one to understand the sign that is made and another one to animate a sign gesture. The first model is based on a Bidirectional LSTM and predicts the name of the gesture. The second model learns to generate sequences of 3D hand keypoints from only the name of the sign, its a is a label-conditioned LSTM network trained with teacher forcing, which helps it learn stable movements. The dataset is made using MediaPipe Hands [4], which tracks 21 landmarks on one hand in each frame. Each recording is two seconds long and centred relative to the face to make the motion consistent. We apply several data augmentation methods to increase the size of the dataset. A Python animation draws the results as a stick figure.

The dataset consist of only 10 classes to allow a light model to be implemented and a lighter dataset to reduce the training time.

The first model, the predictor, produces only good result for signs who are widely different from the others signs of the set. Similar signs are confused by the model.

The animator/generator model show smooth sign gestures, generated from a label. Which can later be used for sign language avatars.

Introduction

Sign languages use sequences of hand movements, shapes, and positions to communicate meaning. Making smooth sign animations for avatars is difficult because it requires accurate control of hand positions over time. Traditional animation methods need manual work to create every frame, which takes a lot of time and often does not look natural. With the help of deep learning and pose-tracking tools like MediaPipe, we can now capture and generate gesture data smoothly.

The aim of this work is to create a deep neural network that can generate a complete sign sequence just from the name of the sign. This system can be used in avatars to display sign language without the need for motion capture at the moment of use.

Related Work

Recent technical efforts have pushed the limits of avatar realism and linguistic accuracy. DiffSign [5] uses a diffusion-based generative model to retarget human signing poses onto 3D avatars with customizable appearance, producing highly natural animations.

Older gesture animation methods were based on motion capture and manual animation blending like eSIGN project [6]. In sign language technology, datasets such as RWTH-PHOENIX-Weather and ASL Lexicon are used for both recognition and generation tasks.

Many sequence models use teacher forcing to make training more stable. This means that during training, the model sees the correct previous step instead of its own prediction. Bengio et al [1] later improved this with a method called scheduled sampling. Conditional generation, where the model is given extra information like a label, is used in other areas such as gesture from speech or handwriting synthesis.

Pose estimation frameworks like MediaPipe Hands allow researchers to extract accurate keypoint positions without other equipment. This makes it possible and simple to build custom datasets for training gesture models.

Methodology

Dataset

An ASL lexicon easy to use on google colab is not common. So a self made dataset was used. Using MediaPipe to cpature the keypoint of a hand.

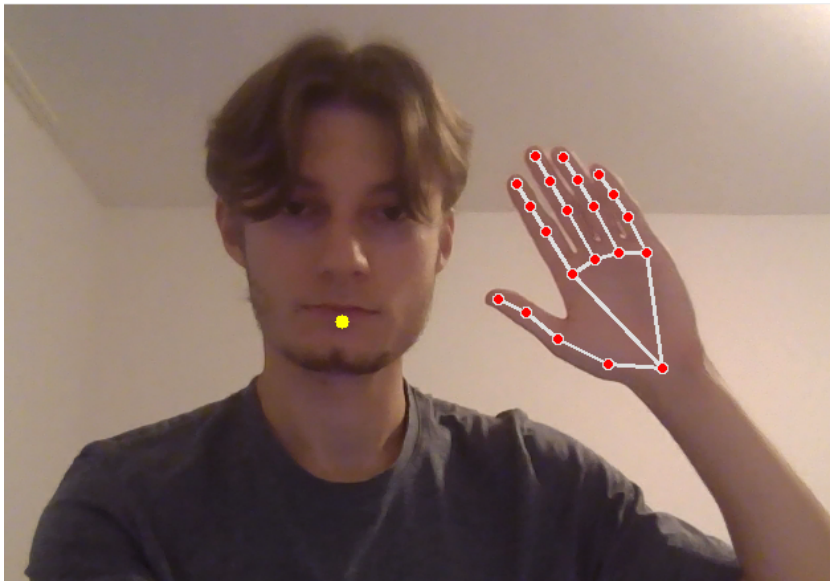

Figure 1: Screenshot of the capture.py program

Each clip is two seconds long, recorded at 30 frames per second, giving 60 frames per sequence. We track 21 landmarks for only hand, each with x, y, z coordinates, so each frame has 63 numbers. The positions are adjusted so they are relative to a point in the head, detected using MediaPipe Face Mesh. This keeps the motion consistent and relative to the head (yellow point on the screenshot). Files are saved as .npy arrays, and the filenames include both the sign label and whether the clip is for training or validation.

We recorded seven clips for each of ten different signs, and two clips per sign for validation.

The dataset was design for the predictor model, with at the beginning 30 classes to predict and not enough data to be trained on. As a result the model overfitted. To resolve this issue, less classes where used (10), more data were captured, augmentation methods were used to increase the size of the dataset and less weight given to the model.

The label of the classes are :
'hello','thanks','no','smart','need','find','fuck','me','home','yes'

Preprocessing

All sequences are resized to 60 frames if needed. We normalize the scale by measuring the distance between the wrist and the base joint of the index finger. The dataset is standardized using the mean and standard deviation from the training set. For the generator model, frames with no hand detected are marked with a mask so they are ignored in the loss function.

Several augmentations are used, such as adding noise, shifting in time, scaling, dropping random frames, and rotating around the z-axis.

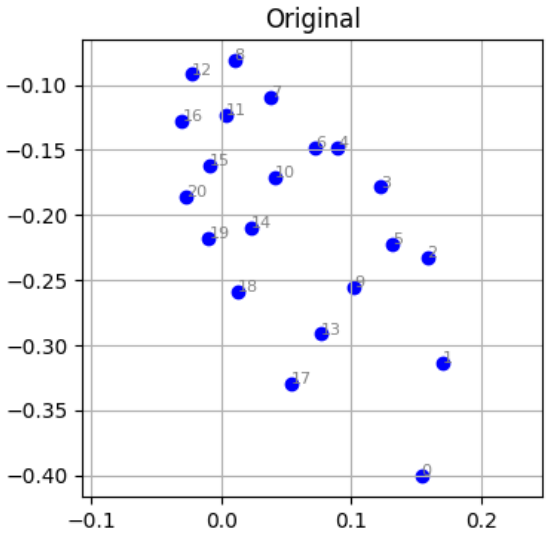

Figure 2: original data

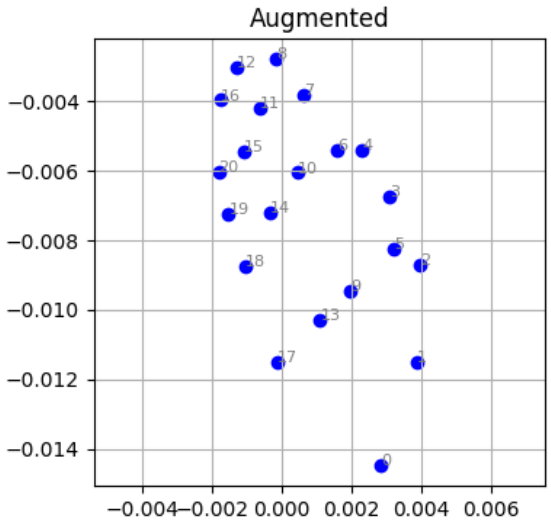

Figure 3: augmented data

There is a slight difference between these two data, in particular a light rotation and some noise applied that slightly modified the original data.

Predictor Model

The Predictor Model is a bidirectional LSTM. LSTM [2] are RNN (recurrent neural network), this type of model posses a memory modified through the input given to the LSTM. So the RNN can understand the input through the sequence. This model is bidirectional, because two LSTM will read the sequence, one from the begining and the other one from the end, the output will be the concatenation of these two LSTM. This design allow the model to remember the begining and the end of the sequence. As a classification task, the last activation function is softmax to have a probability density.

Generator Model

This model is a label-conditioned LSTM generator. The label (sign name) is turned into a learned embedding vector. A time embedding is also added to help the model know the position in the sequence. These are combined with the previous frame and passed through an LSTM with two layers and a hidden size of 256. The output is the next frame's keypoints.

During training we use teacher forcing, meaning the previous frame from the real sequence is given to the model. At inference, the model's own predicted frame is fed back to generate the next step, starting from a zero frame.

Animation

The predicted keypoints are transformed back to their original scale and shown as stick figure animations. The connections between points match the MediaPipe hand model. Animations can be displayed in Jupyter notebooks.

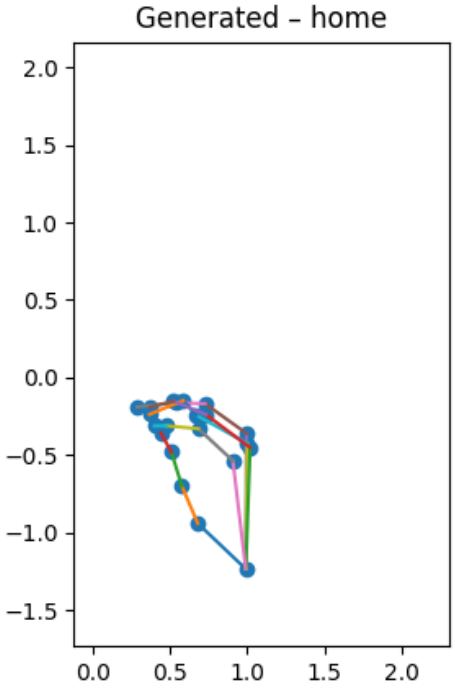

Figure 4: Frame of the sign gesture home

This frame show the keypoints and link between these point, for the whole drawing to look like a hand.

Tools

The library and tools use for this project are in particular **Tensorflow** for the predictor model, **PyTorch**, **IPython** and **Matplotlib** for the animation and the design of the animator model.

Evaluation, Metrics and Validation

For the two model, the validation is based on the part of the dataset which is not use for training.

Two little runner in python were programmed to use and test the different model on the computer instead of in google colab.
"runner_predictor.py" and "runner_generator.py"

For the predictor model, here are some precise information about the accuracy of the model for each sign :

Precision:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1-Score:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

TP (True Positives), FP (False Positives),FN (False Negatives).

	precision	recall	f1-score	support
hello	1.00	1.00	1.00	2
thanks	0.67	1.00	0.80	2
no	0.33	0.50	0.40	2
smart	1.00	1.00	1.00	2
need	0.50	0.50	0.50	2
find	1.00	0.50	0.67	2
fuck	1.00	1.00	1.00	2
me	1.00	0.50	0.67	2
home	0.67	1.00	0.80	2
yes	1.00	0.50	0.67	2
accuracy			0.75	20
macro avg	0.82	0.75	0.75	20
weighted avg	0.82	0.75	0.75	20

Figure 5: Predictor model accuracy for each sign

For these three scores, 1 is the best to get, it corresponds to zero FP and zero FN. The resulting accuracies are not homogeneous, we can expect overconfidence in prediction and error.

Results and discussion

After some test with the program "runner_predictor.py" the model predictor doesn't recognize some sign and is over confident in his error, while other sign are over predicted, such as hello or "fuck", at the expense of the other sign. Because this model is quick to train, we can experiment different training result. There is always some sign who are easily recognize and some other who are totally not.

It is certainly due to the lack of data for the training.

Here are the last Epochs training (train) and validation (val) lossses of the generator model:
Epoch 060 — train 0.07081 — val 0.07614
These two values are low, the animator model generate accurate sign gesture. Indeed, by testing with the runner program, apart from keypoint who are sometimes misplace we understand the gesture as a whole. The short video shown as animation is centred on the sign and not relative to the a keypoint for the head but the sign are different enough for it to be recognize.

People who are used to sign, tend to be quick and to differ from the sign that are teach or that could be find on internet. People have different way to sign the same gesture and to represent this diversity, the dataset need to be build by capturing sign gesture of different person, which is not our case. It is another limitation from the dataset.

Conclusion

Two models were design and trained, one for prediction of text from sign, another one for generation of sign from text. The predictor model is limited by the dataset, but some sign are still recognize unfortunately at the expense of others. Whereas the animator model generate accurate sign, it is not limited by the dataset because overfitting is not a problem in our situation. This project show us that a model can recognize a sign gesture and generate one.

The use of a deep learning model at our scale to generate an animation compared to directly show the capture associated to the sign can be discussed. But this method is a proof of concept for a future model that can realistically generate a gesture. This future model could be applied to animate movement of avatar even if the model was not specifically train for, but trained to generate general realist gesture.

References

[1] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, *Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks*. Advances in Neural Information Processing Systems (NeurIPS), 2015.

[2] S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory*. Neural Computation, 9(8):1735–1780, 1997.

[3] X. Yan, A. M. Dai, and Q. V. Le, *Paired Recurrent Models for End-to-End Handwriting Generation*. Advances in Neural Information Processing Systems (NeurIPS), 2016.

[4] C. Lugaresi, et al., *MediaPipe: A Framework for Building Perception Pipelines*. arXiv:1906.08172, 2019.

[5] Y. Zhu, et al., *DiffSign: Diffusion-based Sign Language Avatar Generation from Human Poses*, arXiv:2412.03878, 2024.

[6] I. Steiner, J. Hanke, T. Heloir, and H. Niemann, *Design, Implementation, and Evaluation of an Avatar for Deaf Signing*, International Conference on Universal Access in Human-Computer Interaction (UAHCI), Springer, pp. 993–1002, 2007. doi:10.1007/978-3-540-73279-2_109