The main things we could have improved! (Currently based on the top 3 results discussions)

Top 1(<https://www.kaggle.com/competitions/asl-fingerspelling/discussion/434485>)

Top 2(<https://www.kaggle.com/competitions/asl-fingerspelling/discussion/434588>)

Top 3 (<https://www.kaggle.com/competitions/asl-fingerspelling/discussion/434393>)

* Increasing the frame\_length, we have limit it to only 128 frames (384, 768, 320)
* All of them normalized the features (We didn’t as it was mentioned that the landmark were already normalized, maybe not efficiently then, and specially pose)
* Their main models imply single encoder-decoder with squeezeformer as encoder and decoder a ‘simple’ 2-layer transformer, Conv2D, use cross entropy loss ; transformer based Decoder superior to a CTC based decoding (1)Use joint CTC +Attention, Conv 1D, Loss = CTC(weight=0.25) + CCE with label smoothing=0.1~0.25(weight=0.75)
* Inference : Replacing positional encoding by Llama attention to 3x faster inference time ; For decoder inference early stopping and past key value caching was used which sped up inference significantly. ` tf-lite inference used a single sample without padding, while model training was performed with time padded mini-batches. To avoid the model learning with pads, and ensure optimal inference runtime, the feature extractor and Macaron structure encoder layers were masked time wise during training. This needed to be manually implemented in pytorch on each layer. This took some effort, but paid off by significantly speeding up inference. As [1st place team](https://www.kaggle.com/competitions/asl-signs/discussion/406684" \t "_blank) in the previous ISL competition explained, this is much easier to do in tensorflow as keras has an off the shelf masking layer.` (1)

If the decoder is an RNN or Transformer, stateful inference(i.e. previous key, value caching with Transformer) can be used to reduce the complexity. In actual implementation with Transformer, it reduced the inference time on the CPU by about 20~30%. (2)

* Replacing corrupted sequence label where we cannot make any predictions by a dummy phrase that will reduce the Levenshtein distance. (~+0.006)
* The top 1 used pytorch for developing and training models, and then manually translated model architecture and ported weigths to tensorflow -> tf-lite. (1) The top 3 used pytorch and converted it to keras using `nobocu` and worked really well. (3)
* The top 3 used tfrecord as input format, and wrote a torch iterable dataset to wrap tfrecord reader.

TOP 2 : (https://www.kaggle.com/competitions/asl-fingerspelling/discussion/434588)

CTC + Attention model joined cf papers: <https://arxiv.org/abs/1609.06773> (**Joint CTC-Attention based End-to-End Speech Recognition using Multi-task Learning)**

; <https://aclanthology.org/P17-1048/> ([Joint CTC/attention decoding for end-to-end speech recognition](https://aclanthology.org/P17-1048.pdf))