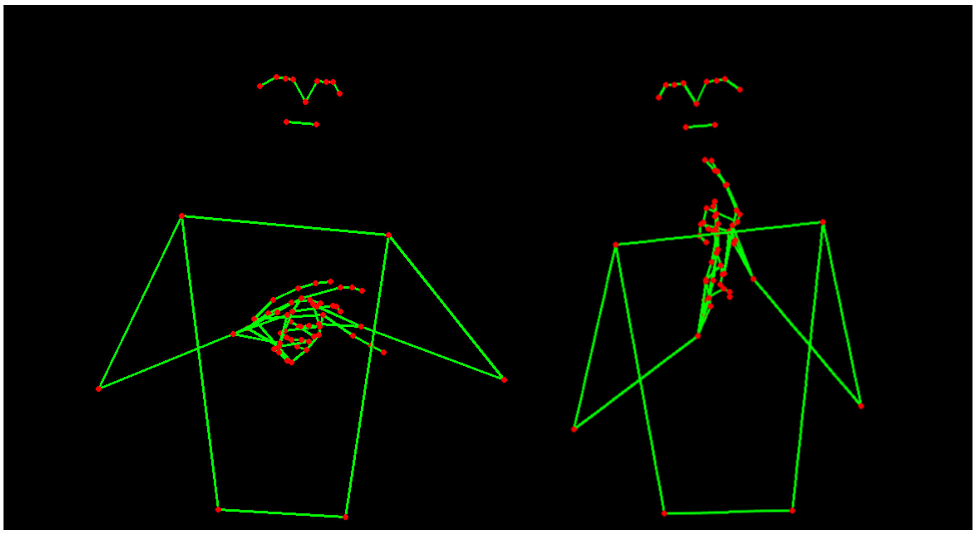
Gesture2Vec - Detecting and recognising unknown gestures for classification purposes



Kevin De Keyser

Master Thesis
September 2021

Supervisors:
Jonas Hein
Prof. Dr. Marc Pollefeys

ETH zürich

Computer Vision
and Geometry Lab

Abstract

In this thesis we propose a novel pipeline for classifying unknown hand gestures. For this purpose, we train a state-of-the-art skeletal action recognition model MS-G3D [Liu et al. 2020b] fitted on a greek sign language dataset [Adaloglou et al. 2020] after reconstructing its skeletons with a tool like MediaPipe [Lugaresi et al. 2019].

Further, we suggest a novel segmentation technique for the unknown gestures by introducing the recently developed Entropic Open-Set loss [Dhamija et al. 2018] to MS-G3D. Lastly, if unknown gestures are detected by the segmentation step, we propose a way to extract embeddings from the model.

We conclude by showing the effectiveness of the techniques, 95% overlapping accuracy rate for the segmentation technique and up to 87% classification accuracy on the embeddings, given 4 samples of unknown gestures per class. Additionally, we propose various ways to improve this pipeline for more error-sensitive applications in the Future Works section.

Acknowledgement

I want to thank my supervisor Jonas Hein for his continued support during the weekly interviews, for giving me very detailed feedback on the structure of the thesis and for enabling this passion-driven MSc. thesis during the COVID lockdown.

Further, I would also like to thank Oswald M. Ralf for connecting me with Jonas and Prof. Dr. Marc Pollefeys for leading the Computer Vision and Geometry Lab in which I completed my thesis.

Next, I want to thank Jan J. A. Schnyder for showing me the field of Open-Set recognition, which found application in my thesis.

Lastly, I want to thank my father, David De Keyser, for financially enabling my pursuit of a Master's degree and being a great moral support during this time.

Contents

List of Figures

List of Tables

1

Introduction

The sole justification for mathematical contemplation is the usefulness of the actions that follow from it, by which I mean the following: Humans can freely control their actions and can bring forth, through cool calculation, experiences which they instinctively desire, but which they cannot bring about by a direct impulse of the will, called a goal.

L.E. J. Brouwer

Throughout human history, gestures were actions of prime importance for human communication. In the service of self-preservation, people subjugated the meaning of these gestures over to the next generation. At lower levels of culture through parent-to-child teaching, and, in addition, at higher levels of culture through institutions, through semiotic literature and now even through datasets and gesture recognition algorithms. Gestures are used to signify social significance through firm handshakes and salutes, to coordinate car or airplane traffic, to communicate with animals, to signify approval through clapping or knocking, to communicate with deaf people through sign languages, or, more relevant today, to signify that you are present in a zoom call, among others.

Today, computers are tasked with prescribing or describing gestures. These gestures can be represented as 3D models, such as skeletal graphs or volumetric meshes, or they can be represented as they appear in 2D images with optional depth mapping.

Most of the work around computational gestures has been done to **prescribe** gestures to robots or computer animation rigs, which is done through forward/inverse kinematics for skeletal data [Paul 1981], mass-spring systems for volumetric data [Tang and Hui 2009] and more advanced techniques such as posture control via PD controllers [Tomei 1991] or operational space control [Khatib 1987].

This thesis focuses on **describing** gestures with computers. Practical applications include controlling an augmented reality headset [Cao et al. 2019], monitoring actions/drowsiness/intent of car drivers through a webcam before they execute their decisions [Molchanov et al. 2016] [Ding], interacting with industrial robots [Lei et al. 2019], sign language recognition [Konstantinidis et al. 2018], playing a game through a webcam or communicating in video conferences.

In order to describe gestures, they first have to be recorded, either visually or mechanically. Visually, gestures are recorded in through optical systems such as standard RGB video, depth videos as provided with the popular Kinect or optical tracking of performers annotated with passive markers. Mechanically, gestures can be recorded through wired gloves (also referred to as datagloves or cybergloves) or less commonly through inertia guided sensors or stretch sensors.

The *video representation* (appearance-based representation) can utilise more information such as shading from the sun or background context. For example, if one tries to classify traffic police hand signals, the behaviour of the cars in the background can be used as an additional clue to get better performance. Background information can, however, also lead to unintended training biases. In the previous example, the traffic gesture system should base predictions primarily around the gestures and not around the behaviour of the cars, for example. Another example is a gesture recognition system that detects some gestures based on whether the user wears gloves when it should not matter.

The *3D skeletal representation* meanwhile gives a more consistent environment, is perspective invariant and is not susceptible to biases based on the background but, conversely, cannot take advantage of background video clues such as colour.

In practice, the data is often converted from one format into the other. For converting appearance-based data to skeletal data, many tools exist. At the time of writing, there are a few popular tools to convert appearance-based data to skeletal data, such as OpenPose [Cao et al. 2018] which achieves near real-time detection with GPU acceleration for 2D skeleton recognition and Google's MediaPipe [Lugaresi et al. 2019] which manages to create real-time 3D skeletons using only CPU power, even on a smartphone. Furthermore, Vibe [Maro et al. 2020] or HRNet [Sun et al. 2019] provide state-of-the-art alternatives trading in power and real-time capability for higher accuracy.

Research in the field of recognising and classifying skeletal movements is called action recognition. It is done primarily on coarse human skeleton models recorded through a system like the Kinect, which is then post-processed to get accurate skeletal representations. Popular datasets in skeletal action recognition include [Shahroudy et al. 2016a] [Liu et al. 2020a] [kin]. The research in this field carries over well to gesture recognition and is discussed in more detail in the Related Work Section.

1.1. Problem statement

We are interested in providing a solution for near real-time gesture recognition and segmentation of gestures that are unknown at training time. As a prerequisite, we are interested in segmenting the unknown gestures from meaningless movements first. Whenever this segmentation is successful, we are then interested in classifying these unknown meaningful gestures.

Such a system would allow users to create their own gestures on the fly and incorporate them with industrial or gaming applications. A good example of real-time user-created commands in video games is Nintendogs commercial voice commands [Nintendo 2005]. We target two main applications in this thesis, detecting gestures from headset mounted cameras like the HoloLens and detecting gestures through webcam support. The skeletal representation of gestures is more suitable for this task as it is perspective invariant and, in addition, uses fewer data and has fast support through tools like MediaPipe [Lugaresi et al. 2019]. The data for these problems feature upper-body skeletons only, as these can be derived from a webcam or derived facing away from a headset such as the HoloLens.

2

Related Work

2.1. Early work

A good survey of early computer-based gesture recognition systems can be found in [Sturman and Zeltzer 1994] and [Richard 1993]. The latter is the first survey to taxonomise gesture recognition systems into template-based approaches, statistics-based approaches and neural-network-based approaches, referred to more generally as feature-extraction approaches here. In addition, it mentions the similarities between handwriting detection; however, due to the 3-dimensional nature of hand gestures, hand gesture detection is a significantly more challenging problem.

2.1.1. Template-based approaches

The first attempts of computerised hand gesture recognition were straightforward and often used linear classifiers together with lookup tables for each joint. We call these template-based approaches. The Nintendo PowerGlove, for example, had a sensor with low accuracy, effectively mapping each finger to roughly four states (completely retracted to fully extended with two additional in-between states). Programming gestures could be made by mapping all gestures to an exhaustive table of finger combinations. A similar approach was used by the MIT media glove, only that they detected the skeleton through LED markings on a glove which function as video annotations [Sturman and Zeltzer 1994]. [Grimes 1981] of Bell Lab Technologies was the first to try to recognise American Deaf sign language with a wired glove and attempted to do this with hard-wired circuitry.

These can be effective on a small number of gestures. For example, [Tolentino 2014] achieved a 93% accuracy for distinguishing the 26 letters of ASL. This approach becomes infeasible for more gestures, as all the templates have to be manually encoded instead of being learnt.

2.1.2. Feature-extraction based approaches

Kramer, who commercialised the CyberGlove [Kramer et al. 1991] in 1990 for research purposes, had the first feature-based approach to gesture recognition, where each degree-of-freedom (DOF) is linked to a dimension of a vector space. He used a "bayesian decision rule-based pattern matching" approach according to a Microsoft survey [Sturman and Zeltzer 1994]. [Sturman and Zeltzer 1994] mentions two early vector-based approaches, one using principal component analysis by ATR Research Labs in Japan and the first neural network-based approach by Fels, who used a 3-stage forward neural network to detect the hands [Fels and Hinton 1993]. In particular, Fels recorded the joint positions and the movement direction of the joints to create a total of 203 gesture-to-word vocabulary and achieved a high accuracy of 94% with this approach on 7822 training examples. In addition, Fels normalised the gesture based on the minimum and maximum flex angles of the gestures.

Until the late 2000s, early 2010s, the leading choice of model for gesture recognition were hidden Markov model-based due to the insensitivity to temporal variation and computational performance, as shown in surveys of human gesture recognition [Mitra and Acharya 2007] [Cheng et al. 2015]. One of the first HMM approaches came from [Yang and Xu 1994] to recognise nine gestures. [Wilson and Bobick 1999] merges hidden Markov based models with neural networks where each state has a neural network estimating the variation of the Gaussian probability density of its output.

Other approaches in this time included multi-class support vector machines [Dardas and Georganas 2011], Sugeno type fuzzy inference system [Kishore and Kumar 2012], Kalman filters [Jeong et al. 2002], Haar wavelet based techniques [Rautaray 2012], ICP matching [Trindade et al. 2012], Genetic algorithms [Ghotkar et al. 2012] and more, see [Sarkar et al. 2013] which gives a great overview of methods pre 2013.

2.2. Recent work / Neural network-based approaches

With increasing computational power, it became possible to differentiate between a larger number of classes. Larger datasets like [Shahroudy et al. 2016a], [Liu et al. 2020a] and skeletal-based [kin] enabled benchmarking of skeletal-based methods for action recognition. These datasets worked on a coarse representation of the entire human body skeleton (not just gestures) and are fairly noisy. Kinetics 400 is a dataset published by DeepMind which has been processed with tools such as OpenPose [Cao et al. 2018], MediaPipe [Lugaresi et al. 2019], Vibe [Maro et al. 2020] or HRNet [Sun et al. 2019]. MediaPipe and OpenPose are the projects with the largest support behind them. They are readily available as they work out of the box without much configuration required and can be run without a GPU, albeit slower. Both are used in this thesis.

Video-based classification techniques are also used, taking advantage of recent breakthroughs in video recognition, such as [Duan et al. 2021] which uses 3D-CNNs based on SlowFast [Feichtenhofer et al. 2019], by rendering the skeletons as heatmap videos. This approach achieved equal performance to [Liu et al. 2020b].

The most faithful and complete benchmark at the time of writing can be found at [Niais 2021] which base their ranking on the [Shahroudy et al. 2016a] and [Liu et al. 2020a] dataset, see Figure 2.1.

Since actions, as well as hand gestures, are time-based data, popular techniques from sentence prediction have been repurposed for gesture recognition, such as the Recursive Neural Networks (RNN), the Long-short term memory (LSTM) architecture, and the attention-mechanism [Si et al. 2019].

Spatio-temporal graph convolutional neural networks and video-based 3D-CNNs outperform all other methods in the field at the time of writing. Our thesis builds on spatio-temporal graph convolutional neural networks, specifically on MS-G3D [Liu et al. 2020b]. However, our segmentation technique can be used on all methods that use a Cross-Entropy loss in the final layer.

2.3. Spatio-temporal graph convolutional neural networks

Due to the curse of dimensionality, learning on high dimensional data is challenging. A sequence of skeletal data can quickly have high dimensionality. In our case, we use 75 joints each containing 3 spatial dimensions/channels per frame, and although there is no fixed frame count, the average number of frames per gesture is 17 which gives a $17 \cdot 75 \cdot 3 = 3825$ dimensional data vector.

In the geometric deep learning paper [Bronstein et al. 2021] mention that the best model to train some data $\mathbf{X}$ without imbuing geometric or ordinal structure on the dimensions would have to have the following form introduced by the deep set model in [Zaheer et al. 2018]:

$$\begin{aligned} f(\mathbf{X}) = \phi\left(\bigoplus_{x_i \in \mathbf{X}} \psi(\mathbf{x}_i)\right) &, \text{ where } \phi, \psi \text{ are generic learnable functions,} \\ &\text{and } \bigoplus \text{ is an aggregation function like sum, average, min, max, etc.} \\ &\text{and } f \text{ satisfies } \forall \mathbf{P} \in S_n . f(\mathbf{PX}) = \mathbf{P} f(\mathbf{X}) \\ &\quad \text{where } S_n \text{ is the set of all permutations of size n.} \end{aligned} \tag{2.1}$$

[Bronstein et al. 2021] provide a general definition for graph neural networks, where geometric structure is imbued to the model by an adjacency matrix $\mathbf{A} \in \mathcal{G}$ in addition to the data $\mathbf{X}$, and instead of applying ψ to every node in isolation it is a binary function applied to the node and its neighbourhood. The neighbourhood of a node v is defined as $\mathcal{N}(v) := \{w | (v, w) \in \mathbf{A}\}$

We define F to be the entire layer of the graph neural network which has to satisfy permutation-

Year	Model name	Citation	Cross-Subject score	Cross-View score
2014	Lie Group	[Vemulapalli et al. 2014]	50.1	52.8
2015	H-RNN	[Du et al. 2015]	59.1	64.0
2016	Part-aware LSTM	[Shahroudy et al. 2016b]	62.9	70.3
2016	Trust Gate ST-LSTM	[Liu et al. 2016]	69.2	77.7
2017	Two-stream RNN	[Wang and Wang 2017]	71.3	79.5
2017	STA-LSTM	[Song et al. 2016]	73.4	81.2
2017	Ensemble TS-LSTM	[Lee et al. 2017]	74.6	81.3
2017	Visualization CNN	[Liu et al. 2017]	76.0	82.6
2017	C-CNN + MTLN	[Ke et al. 2017]	79.6	84.8
2017	Temporal Conv	[Kim and Reiter 2017]	74.3	83.1
2017	VA-LSTM	[Zhang et al. 2017]	79.4	87.6
2018	Beyond Joints	[Wang and Wang 2018]	79.5	87.6
2018	ST-GCN	[Yan et al. 2018]	81.5	88.3
2018	DPRL	[Tang et al. 2018]	83.5	89.8
2019	Motif-STGCN	[Wen et al. 2019]	84.2	90.2
2018	HCN	[Li et al. 2018]	86.5	91.1
2018	SR-TSL	[Si et al. 2018]	84.8	92.4
2018	MAN	[Xie et al. 2018]	82.7	93.2
2019	RA-GCN	[Song et al. 2019]	85.9	93.5
2019	DenseIndRNN	[Li et al. 2020]	86.7	93.7
2018	PB-GCN	[Thakkar 2018]	87.5	93.2
2019	AS-GCN	[Li et al. 2019]	86.8	94.2
2019	VA-NN (fusion)	[Zhang et al. 2019]	89.4	95.0
2019	AGC-LSTM (Joint&Part)	[Si et al. 2019]	89.2	95.0
2019	2s-AGCN	[Shi et al. 2019b]	88.5	95.1
2020	SGN	[Zhang et al. 2020]	89.0	94.5
2020	GCN-NAS	[Peng et al. 2019]	89.4	95.7
2019	2s-SDGCN	[Wu et al. 2019]	89.6	95.7
2019	DGNN	[Shi et al. 2019a]	89.9	96.1
2020	MV-IGNET	[Wang et al. 2020]	89.2	96.3
2020	4s Shift-GCN	[Cheng et al. 2020b]	90.7	96.5
2020	DecoupleGCN-DropGraph	[Cheng et al. 2020a]	90.8	96.6
2020	PA-ResGCN-B19	[Song et al. 2020]	90.9	96.0
2020	MS-G3D	[Liu et al. 2020b]	91.5	96.2
2021	EfficientGCN-B4	[Song et al. 2021]	91.7	95.7
2021	CTR-GCN	[Chen et al. 2021]	92.4	96.8

Figure 2.1.: *Benchmark for Skeletal Action Recognition on the NTU-60 [Shahroudy et al. 2016a] dataset by [Niais 2021].*

equivariance, ensuring no information is given about the order of the data:

$$F(\mathbf{X}, \mathbf{A}) := \begin{bmatrix} f(\mathbf{x}_1, \mathbf{A}) \\ \vdots \\ f(\mathbf{x}_N, \mathbf{A}) \end{bmatrix}, \quad \text{where } F \text{ satisfies } \forall \mathbf{A} \in \mathcal{G}.\forall \mathbf{P} \in \mathcal{S}_n.\mathbf{P}F(\mathbf{X}, \mathbf{A}) = F(\mathbf{PX}, \mathbf{PAP}^T) \tag{2.2}$$

This is a general definition of a graph neural network, [Bronstein et al. 2021] calls them message-passing GNNs:

$$f(x_u, A) = \phi\left(x_u, \bigoplus_{v \in \mathcal{N}(u)} \psi(x_u, x_v)\right), \text{where } \phi \text{ and } \psi \text{ are generic learnable functions,}$$
$$\text{and } \bigoplus \text{ is an aggregation function like sum, maximum, etc.} \tag{2.3}$$

Attention-based graph neural network layers are a specific subset of GNN layers, with $\psi(x_u, x_v)$ being split into a product $a(x_u, x_v) \cdot \psi(x_v)$:

$$f(x_u, A) = \phi\left(x_u, \bigoplus_{v \in \mathcal{N}(u)} a(x_u, x_v) \cdot \psi(x_v)\right), \text{with the same properties.} \tag{2.4}$$

Convolutional graph neural network layers are a subset of attention-based GNN layers, with $a(x_u, x_v) := \mathbf{w}_{uv} \cdot \psi(x_v)$:

$$f(x_u, A) = \phi\left(x_u, \bigoplus_{v \in \mathcal{N}(u)} \mathbf{w}_{uv} \cdot \psi(x_v)\right), \text{with the same properties.} \tag{2.5}$$

Image-based convolutional neural network layers are a grid-based subset of graph convolutional neural network layers with $A = \mathcal{G}_{\text{grid}}$, where the weights are denoted as $\mathbf{W} \in \mathbb{R}^{W \times H \times C \times f_l}$ with f_l being the number of filters in the l-th layer.

$$f(x_{i,j,c}) = \phi\left(\sum_{m=1}^{H}\sum_{n=1}^{W}\sum_{c=1}^{C} x_{i+m,j+n,c} \cdot \mathbf{w}_{m,n,c}\right), \text{where } \phi \text{ is usually ReLU activation function} \tag{2.6}$$

Time convolutional neural network layers are a segment-based subset of grid-based graph convolutional neural network layers, meaning they are simply 1D convolutional neural networks, where the weights are $\mathbf{W} \in \mathbb{R}^{T \times C \times f_l}$ with f_l being the number of filters in the l-th layer.

$$f(x_{i,c}) = \phi\left(\sum_{t=1}^{T}\sum_{c=1}^{C} x_{i+t,c} \cdot \mathbf{w}_{t,c}\right), \text{ where } \phi \text{ is usually the ReLU activation function} \quad (2.7)$$

These convolutional neural networks are multi-scale, meaning they are interleaved by coarsening/pooling layers on the grid structure: $\text{CNN}(x) = Pool(F(x))$ and perform a residual step in addition: $\text{ResNet}(x) = Pool(x + F(x))$. In graph convolutional neural networks applied to action recognition, the neighbourhoods of the skeletal input sequence can be defined with different combinations of spatial, temporal and spatiotemporal edges, see Figure 2.2. TCN [Kim and Reiter 2017] only applies a standard temporal convolution with no graph structure, while ST-GCN [Yan et al. 2018] adds spatial convolutions before applying temporal convolutions and MS-G3D combines these into a large spatiotemporal convolution, where the skeletons are time-wise fully connected, meaning each node belonging to frame t is connected to all other nodes belonging to other frames.

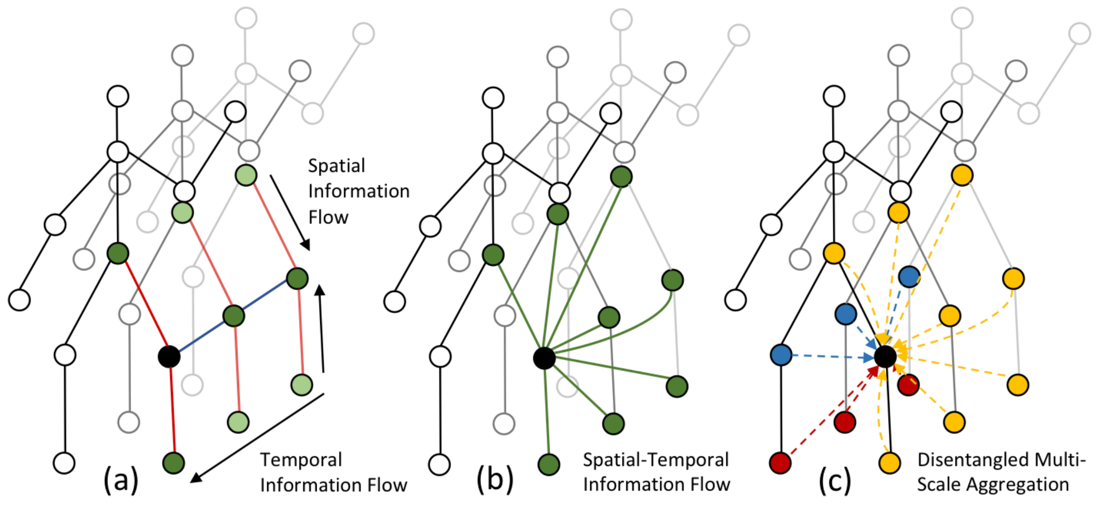

Spatial Information Flow
Temporal Information Flow
(a)
(b)
Spatial-Temporal Information Flow
(c)
Disentangled Multi-Scale Aggregation

Figure 2.2.: *Possible geometric structures of the skeletal data for the use in spatial graph convolutional neural networks. In the left illustration, spatial and temporal structure is separated as in ST-GCN [Yan et al. 2018]. The center illustrates the spatio-temporal structure that MS-G3D [Liu et al. 2020b] uses and the right illustration partitions the nodes by distance. The graphics were taken from [Liu et al. 2020b]*

In this section, we explain TCN [Kim and Reiter 2017] introducing temporal convolutions to this problem, ST-GCN [Yan et al. 2018] introducing spatial convolutional neural networks to this problem, AS-GCN [Li et al. 2019] which introduces larger neighbourhoods to spatial convolutional neural networks, and lastly, MS-G3D [Liu et al. 2020b] which introduces spatio-temporal convolutional neural networks to this problem.

2.3.1. TCN

TCN/TempConv [Kim and Reiter 2017] is a residual temporal convolutional neural network. The entire skeleton is interpreted as a vector with each spatial coordinate of a joint modelling a separate channel with no spatial information in-between them, and no normalisation is applied on the coordinates, see Figure 2.3.

$$\begin{aligned} f(x_{i,c}) &:= \sigma\left(\sum_{t=1}^{T}\sum_{c=1}^{C} x_{i+t,c}\cdot \mathbf{w}_{t,c}\right) \\ F(\mathbf{X}) &:= \sigma\left(\mathbf{X}\cdot\mathbf{W}\right)\ , \quad \text{the temporal convolutional layer} \\ TCN(\mathbf{X}) &:= Pool(\mathbf{X}+F(\mathbf{X}))\ , \quad \text{the TCN module with pooling and residual connection} \end{aligned} \tag{2.8}$$

The entire architecture has 10 layers, the first being a temporal convolutional layer with 64 filters/output channels, followed by 9 residual temporal convolutional layers with output channel sizes $[64, 64, 64, 128, 128, 128, 256, 256, 256]$, followed by an average pooling layer, a regular feed-forward layer of size 64 and finally a Softmax layer. On this, the cross-entropy loss is applied for training. See Figure 2.4 for a graphical representation of the model.

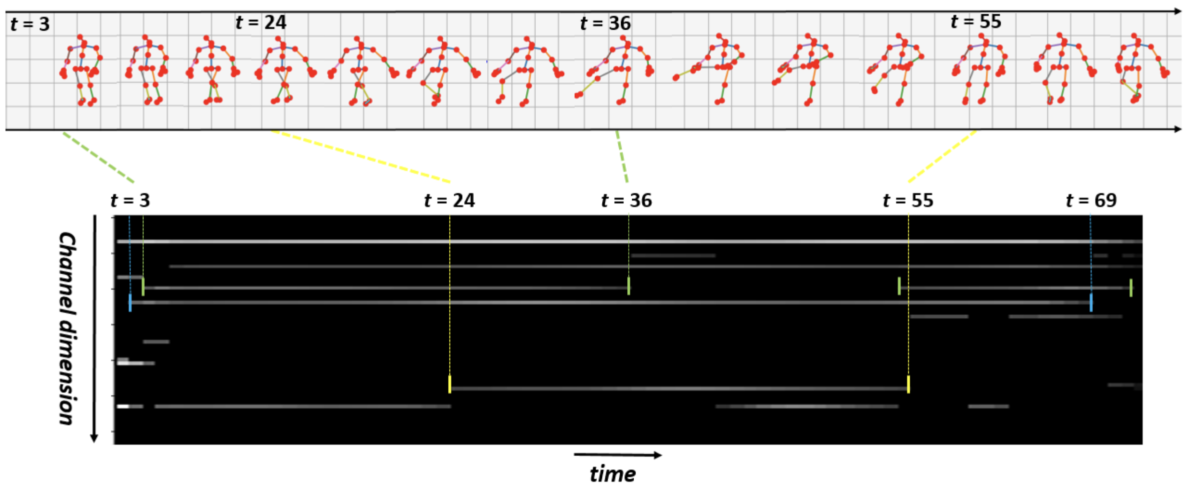


Figure 2.3.: *A visualisation of the TCN channels over time taken from [Kim and Reiter 2017]*

2.3.2. ST-GCN

To explain ST-GCN, we use the notation introduced in MS-G3D [Liu et al. 2020b]. The human skeleton is given as an adjacency matrix $\mathbf{A} \in \mathcal{G} \subseteq \mathbb{R}^{N\times N}$, where $\mathbf{A}_{i,j} = 1$ if the i-th joint is connected with the j-th joint, otherwise it is 0.

Actions or gestures are defined as $\mathbf{X} \in \mathbb{R}^{T\times N\times C}$, where T is the number of frames in the gesture, N is the number of joints in the skeleton, and C are the number of dimensions/channels. In the first layer, C represents the number of spatial dimensions of the given skeleton sequence. Skeleton sequences generated by MediaPipe create two-dimensional skeletons $C^{(0)} = 2$ while OpenPose generates three-dimensional skeletons $C^{(0)} = 3$. We used MediaPipe as it was the faster framework and resulted in greater classification performance. Further, we denote $\mathbf{X}_t =$

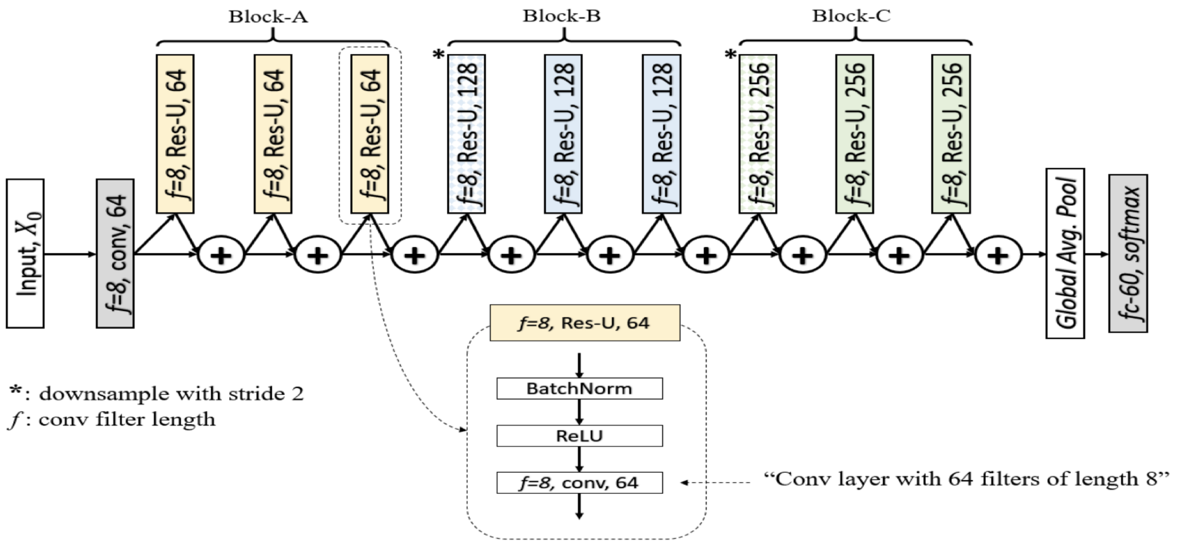


Figure 2.4.: *The TCN architecture visualised taken from [Kim and Reiter 2017]*

$\mathbf{X}_{t,:,:}$ as the skeleton features at time t and $\mathbf{x}_{t,n} = \mathbf{X}_{t,n,:}$ as the C dimensional feature vector for node v_n at time t.

In the ST-GCN model, spatial convolutions are applied first, followed by temporal convolutions; see the left image of Figure 2.2. Hence our graph $\mathcal{A}$ model only the spatial relationship of joints at frame t (without time edges).

We define the neighbour set of node v_{ti} as $B(v_{ti}) := \{v_{tj} \,|\, d(v_{tj}, v_{ti}) \leq D\}$ for a given maximal distance D. d is the edge distance between two vertices. In ST-GCN $D = 1$ is used. With $D = 1$ the neighbourhood can also be represented as the adjacency matrix $\widetilde{A} = A + I$, containing all direct neighbours and itself.

ST-GCN suggests three ways of partitioning the neighbourhoods, namely uni-labelling, distance partitioning and spatial configuration partitioning. See Figure 2.5 for a visualisation of the partitioning schemes. The partitioning schemes are defined as follows:

- In uni-labelling, each neighbour of x_{ti} belongs to the same partition.
- In distance partitioning, the neighbours are partitioned by distance. Since $D = 1$, this leads to a distance 0 label containing x_{ti} and a distance 1 label containing all direct neighbours of x_{ti}.
- In spatial configuration partitioning, the distance 0 neighbour x_{ti} belongs to its own partition, and the distance one neighbours are partitioned by their orientation to the gravity centre of the skeleton, either facing it (centripetal partition) or facing away from it (centrifugal partition).

Each partition of the neighbourhood x_{ti} shares a learnable weight vector $\mathbf{w}_i \in \mathbb{R}^C$ capturing important features of that partition. The output value of a single spatial convolutional layer at

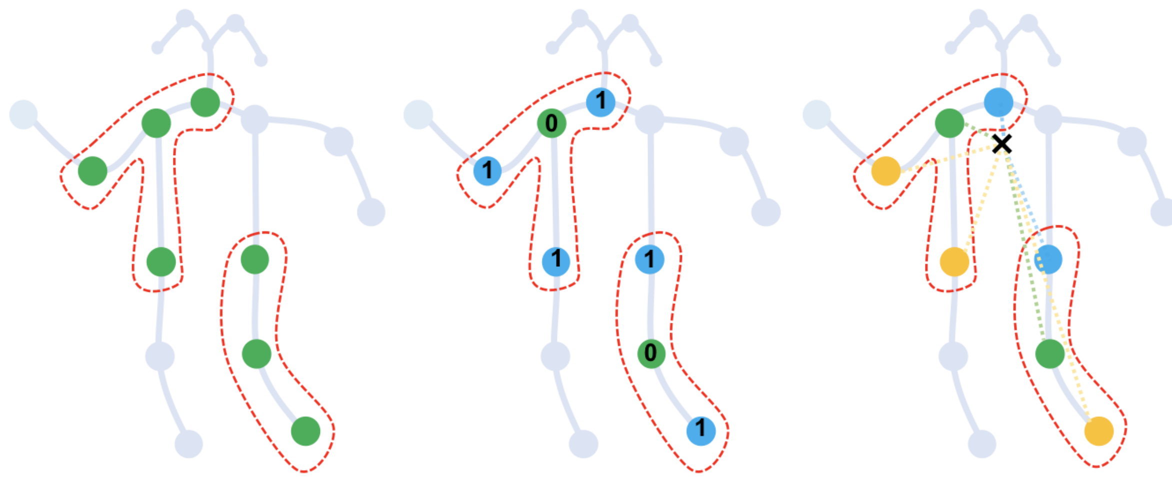


Figure 2.5.: *On the left is the uni-labelling partitioning scheme, in the middle is the distance partitioning scheme with $D = 1$ and on the right is the spatial configuration partitioning. The visualisation has been taken from the ST-GCN paper [Yan et al. 2018]*

node v_{ti} using the uni-labelling partitioning method is defined as follows:

$$f(\mathbf{x}_{ti}, A) = \sigma\left(\sum_{v_{tj} \in B(v_{ti})} \frac{1}{|B(v_{ti})|} \mathbf{x}_{tj} \cdot \mathbf{w}_i\right) \tag{2.9}$$

We define the following matrices to rewrite the expression in a concise form:

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_N \end{bmatrix} \in \mathbb{R}^{N \times C}$$, representing the learnable weights for each node and channel.

$$F(\mathbf{X}_t) = \begin{bmatrix} f(\mathbf{x}_{t1}) \\ \vdots \\ f(\mathbf{x}_{tN}) \end{bmatrix} \in \mathbb{R}^{N \times C}$$, the output of the spatial convolutional layer for every node and channel.

Due to [Kipf and Welling 2017] we can rewrite $F(\mathbf{X}_t)$ as:

$$F(\mathbf{X}_t) = \sigma\left(\widetilde{\mathbf{D}}^{-\frac{1}{2}} \widetilde{\mathbf{A}} \widetilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{X}_t \mathbf{W}\right), \quad \text{where } \widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I} \text{ (distance } \leq 1 \text{ neighbourhood)}$$
$$\widetilde{\mathbf{D}} \text{ is the diagonal degree matrix of } \widetilde{\mathbf{A}} \text{ with} \tag{2.10}$$
$$\widetilde{\mathbf{D}}_{ii} = \sum_j \widetilde{\mathbf{A}}_{ij} = \sum_j \mathbf{A}_{ij} + \mathbf{I}_{ij}$$

In general, we denote the P partitions of the learnable weights by a superscript $\mathbf{w}^{(p)}$ and $\mathbf{W}^{(p)}$ respectively.

For example, the output layer of the spatial convolutional layer for distance partitioning is:

$$F(\mathbf{X}_t) = \sigma\left(\sum_{p\in\mathcal{P}} \mathbf{D}^{(p)^{-\frac{1}{2}}}\mathbf{A}^{(p)}\mathbf{D}^{(p)^{-\frac{1}{2}}}\mathbf{X}_t\mathbf{W}^{(p)}\right), \text{ where } \mathbf{A}^{(0)} = I$$
$$\mathbf{A}^{(1)} = A$$
$$\mathbf{D}^{(p)} \text{ is the diagonal degree matrix of } \mathbf{A}^{(p)} \text{ with}$$
$$\mathbf{D}^{(p)}_{ii} = \sum_j \mathbf{A}^{(p)}_{ij} \tag{2.11}$$

Lastly, to capture edge weights in ST-GCN, another learnable matrix $\mathbf{M}^{(p)}$ is introduced, scaling the contribution of a nodes feature to its neighbouring nodes. Each partition gets its own learnable matrix. The final ST-GCN spatial convolutional layer looks as follows:

$$F(\mathbf{X}_t) = \sigma\left(\sum_{p\in\mathcal{P}} \mathbf{M}^{(p)} \circ \mathbf{D}^{(p)^{-\frac{1}{2}}}\mathbf{A}^{(p)}\mathbf{D}^{(p)^{-\frac{1}{2}}}\mathbf{X}_t\mathbf{W}^{(p)}\right), \text{ where } \circ \text{ denotes the Hadamard product.} \tag{2.12}$$

The temporal convolution is now applied on the output $F(\mathbf{X})$ in the same way as TCN [Kim and Reiter 2017] with an additional residual connection. In the 4th and 7th layer an additional pooling layer is applied. This forms the ST-GCN module:

$$STGCN(\mathbf{X}) = Pool(\mathbf{X} + TCN(F(\mathbf{X}))) \tag{2.13}$$

The ST-GCN module is repeated 9 times with channel sizes $[64, 64, 64,\ 128, 128,\ 128, 256, 256, 256]$, after which a global average pooling layer of size 256 is applied, after which the final Softmax layer is applied.

2.3.3. AS-GCN and MS-G3D

AS-GCN [Li et al. 2019] introduces larger distances $D > 1$ in the neighbourhood, resulting in higher-order polynomials of the adjacency matrix. The distance partitioning concept can be extended to arbitrary sized distances. The neighbourhoods of partition p are defined as $\mathcal{B}^{(p)}(v_i) = \{v_j | d(v_j) = p\}$ and are encoded by the length-p adjacency matrix defined as $\mathbf{A}^{(p)} := \mathbf{A}^p$. The problem with this approach is that $\mathbf{A}^p_{ij} = \mathbf{A}^p_{ji}$ equals the number of paths between vertex i and vertex j of size k. The number of paths rise exponentially as k grows. $\mathbf{D}^{(p)^{-\frac{1}{2}}}\mathbf{A}^{(p)}\mathbf{D}^{(p)^{-\frac{1}{2}}}\mathbf{X}_t$ performs a weighting based on the number of paths which is unintuitive, when instead, as noted by MS-G3D, it could perform a weighting based only on the connectivity and distance.

MS-G3D solves this by defining $\mathbf{A}^{(p)}$ to be the following binary matrix:

$$\mathbf{A}_{ij}^{(p)} := \begin{cases} 1, & \text{if } d(v_i, v_j) = p \\ 1, & \text{if } i = j \\ 0, & \text{else} \end{cases} \tag{2.14}$$

or equivalently,

$$\mathbf{A}^{(p)} := \mathbf{I} + [\mathbf{A}^p \geq 1] = \mathbf{I} + [(\mathbf{A} + \mathbf{I})^p \geq 1] - \left[(\mathbf{A} + \mathbf{I})^{p-1} \geq 1\right] \tag{2.15}$$

MS-G3D merges spatial and temporal convolutions into one by convolving on the graph connecting all the skeletons in a sliding window of size τ, not to be confused with the length of the gesture T. This is done by defining new features at frame t $[\mathcal{X}_{(\tau)}]_t$ containing not only the features $\mathbf{X}_t$ of the skeleton at frame t, but also the features in the sliding window surrounding it: $\mathbf{X}_{t-\tau/2}, \ldots, \mathbf{X}_{t-1}, \mathbf{X}_{t+1}, \ldots, \mathbf{X}_{\tau/2}$ and zero padding values outside the range $\mathbf{X}_{t<0 \vee t \geq T} = 0$.

The structure of the spatiotemporal is defined by the adjacency matrix $\mathcal{A}_{(\tau)}^{(p)}$ connecting all skeletons densely in the time frame of τ and its corresponding diagonal degree matrix $\mathcal{D}_{(\tau)}^{(p)}$. Formally:

$$\mathcal{A}_{(\tau)}^{(p)} := \begin{bmatrix} \mathbf{A}^{(p)} & \ldots & \mathbf{A}^{(p)} \\ \vdots & \ddots & \vdots \\ \mathbf{A}^{(p)} & \ldots & \mathbf{A}^{(p)} \end{bmatrix} \in \mathbb{R}^{\tau N \times \tau N}, \; [\mathcal{D}_{(\tau)}^{(p)}]_{ii} := \sum_j [\mathcal{A}_{(\tau)}^{(p)}]_{ij}, \quad [\mathcal{X}_{(\tau)}]_t := \begin{bmatrix} \mathbf{X}_{t-\tau/2} \\ \vdots \\ \mathbf{X}_{t+\tau/2} \end{bmatrix} \in \mathbb{R}^{T \times \tau N \times C} \tag{2.16}$$

Note that a dilation factor $d \in \mathbb{N}$ can be introduced to the sliding window. In that case $[\mathcal{X}_{(\tau)}]_t$ is modified to be:

$$[\mathcal{X}_{(\tau)}]_t := \begin{bmatrix} \mathbf{X}_{t-d\cdot(\tau/2)} \\ \mathbf{X}_{t-d\cdot(\tau/2-1)} \\ \vdots \\ \mathbf{X}_{t+d\cdot(\tau/2-1)} \\ \mathbf{X}_{t+d\cdot(\tau/2)} \end{bmatrix} \in \mathbb{R}^{T \times \tau N \times C} \tag{2.17}$$

Furthermore, MS-G3D does not use the learnable edge weight matrices $\mathbf{M}^{(p)}$ but instead, adds the learnable edge weight matrices $\mathcal{A}_{\text{res}}{}_{(\tau)}^{(p)}$ directly to $\mathcal{A}_{(\tau)}^{(p)}$ with a similar effect.

The new updated formula is the MS-G3D layer. After specifying τ (the sliding window size) and d (the dilation factor) the equation is as follows. If d is not specified it is assumed to be 1.

$$\text{MS-G3D}(\mathbf{X}, \tau, d)_t := F(\mathbf{X}_t) := \sigma\left(\sum_{p=1}^{P} \mathcal{D}_{(\tau)}^{(p)^{-\frac{1}{2}}} \left(\mathcal{A}_{(\tau)}^{(p)} + \mathcal{A}_{\text{res}}{}_{(\tau)}^{(p)}\right) \mathcal{D}_{(\tau)}^{(p)^{-\frac{1}{2}}} [\mathcal{X}_{(\tau)}]_t \mathbf{W}^{(p)}\right) \tag{2.18}$$

The τ sized-window does not range over the entire input skeleton tensor $\mathbf{X} \in \mathbb{R}^{T \times N \times C}$. For this reason, information shared over distances longer than τ have to be exchanged. This is done by collecting the output of the MS-G3D$(\mathbf{X}, \tau, d)$ function and feeding it into a temporal convolution unit just like the models before it, in this case, it is referred to as MS-TCN. Note that the temporal convolution module MS-TCN is always applied, even if $\tau = T$. MS-TCN consists of parallel standard temporal convolutions with dilations 1,2,3 and 4, in addition to a max-pooling layer and a simple 1x1 convolution. The output of these parallel convolutions gets concatenated, and a residual connection of a 1x1 convolution of stride 2 is added. The network architecture is visualised in Figure 2.6.

Together, applying MS-G3D and then MS-TCN is referred to as an STGC module which is then repeated r times. The STGC module can contain multiple MS-G3D modules in parallel if accuracy is preferred over performance. Also, a standard spatial skeleton convolution with no window ($\tau = 1$) is applied in the factorised pathway with a corresponding temporal convolution. For a visualisation, see Figure 2.6. The standard settings, which we borrowed, with $r = 3$ concatenated STGC modules with output feature channel sizes of $96, 192$ and 384, respectively. At the start of each STGC module, the temporal dimension is downsampled with stride 2, except for the first STGC module. Furthermore, at the end of each STGC module, except for the last, batch normalisation is applied. After the final STGC module, a global average pooling layer is applied, after which a single feedforward layer is applied. Then a cross-entropy loss is applied, comparing the predicted output after applying the Softmax function with the true labels.

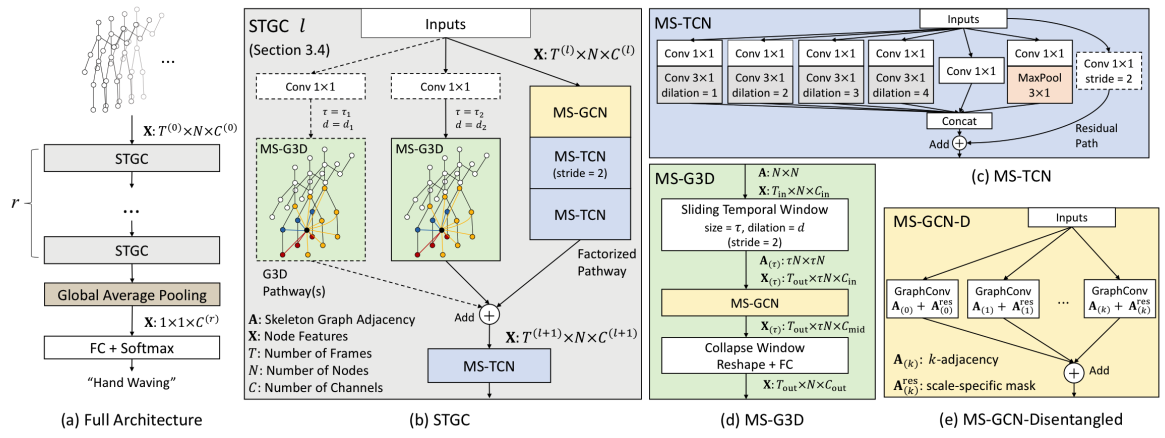


Figure 2.6.: *A visualisation of the full MS-G3D architecture and its submodules taken from [Liu et al. 2020b]*

2.4. Entropic Open-Set Loss

All positive examples are alike; each negative example is negative in its own way.

[Zhou and Huang 2001]

2.4.1. Softmax loss

In recognition tasks, the Softmax layer, sometimes hidden in the cross-entropy loss, is commonly used in the final step of the neural network to create a vector of probability values. It is also used in MS-G3D.

The k-th entry of the output vector of the Softmax layer is trained to be 1.0 if the input data represents the k-th gesture. More specifically, it is trained to give the probability that the input data represents the k-th gesture under the assumption that it is one of the $N = |\mathcal{C}_{\text{gestures}}|$ gestures of the Softmax function. This property is enforced by the fact that the sum of the vector entries/probabilities have to add up to one and ensures that the neural network does not have to learn this property.

For $c \in \mathcal{C}_{\text{gestures}}$ the Softmax and the probability distributions arising from it are defined as:

$$sm_c(x) := \frac{e^{x_c}}{\sum_{c' \in \mathcal{C}_{\text{gestures}}} e^{x_{c'}}} = P\left[Y = c|x\right] \tag{2.19}$$

Then one can proceed with a maximum likelihood estimation on the Softmax probabilities to get the corresponding cross-entropy loss.

$$\begin{aligned}
L(\theta; \mathcal{D}_{\text{train}}) &= \underset{\theta}{\text{argmax}} \prod_{(x,y)\in\mathcal{D}_{\text{train}}} \prod_{c' \in \mathcal{C}_{\text{gestures}}} P\left[Y = c'|x\right]^{y_{c'}} \\
&= \underset{\theta}{\text{argmax}} \; \log \left(\prod_{(x,y)\in\mathcal{D}_{\text{train}}} \prod_{c' \in \mathcal{C}_{\text{gestures}}} P\left[Y = c'|x\right]^{y_{c'}} \right) \\
&= \underset{\theta}{\text{argmax}} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} \sum_{c' \in \mathcal{C}_{\text{gestures}}} \log \left(P\left[Y = c'|x\right]^{y_{c'}} \right) \\
&= \underset{\theta}{\text{argmax}} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} \sum_{c' \in \mathcal{C}_{\text{gestures}}} y_{c'} \cdot \log P\left[Y = c'|x\right] \\
&= \underset{\theta}{\text{argmax}} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} \sum_{c' \in \mathcal{C}_{\text{gestures}}} y_{c'} \cdot \log sm_{c'}(x) \\
&= \underset{\theta}{\text{argmin}} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} \sum_{c' \in \mathcal{C}_{\text{gestures}}} -y_{c'} \cdot \log sm_{c'}(x) \\
&= \underset{\theta}{\text{argmin}} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} J_{sm}(x; y) \\
&= \underset{\theta}{\text{argmin}} \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} J_{sm}(x; y)
\end{aligned} \tag{2.20}$$

After applying the algebraic manipulation from above, it is now in the standard form on which stochastic gradient descent can be applied. Since our label y is a one-hot encoding we can deduce c from y (the non-zero entry filled with 1.0) and rewrite $J_{sm}(x)$ as:

$$\begin{aligned} J_{sm}(x) &= \sum_{c' \in \mathcal{C}_{\text{gestures}}} -y_{c'} \cdot \log sm_{c'}(x) \\ &= -\log sm_c(x) \\ &= -x_c + \log \left(\sum_{c' \in \mathcal{C}_{\text{gestures}}} e^{x_{c'}} \right) \end{aligned} \tag{2.21}$$

Intuitively, when x_c (the value of the target class before applying the softmax) is significantly larger than the other values, the losses cancel as illustrated by this equation[1]:

$$\log \left(\sum_{c' \in \mathcal{C}_{\text{gestures}}} e^{x_{c'}} \right) \approx \max_{c' \in \mathcal{C}} x_{c'} = x_c \tag{2.22}$$

2.4.2. Entropic Open-Set loss

In this thesis, we are not only interested in distinguishing known classes of gestures. For this purpose, we create the following categories:

- $\mathcal{C}_{\text{known}}$: are meaningful classes which are known at training time. The final layer has size $|\mathcal{C}_{\text{known}}|$ representing probabilities over these known gestures.
- $\mathcal{C}_{\text{unknown}}$: are meaningful classes which are unknown at training time and do not have a class in the final layer representing them.
- $\mathcal{C}_{\text{background}}$: is a catch all class for all meaningless movements, such as the movements of the hands in-between two gestures.

Distinguishing between meaningful gestures, whether known or unknown, is effectively solved with embeddings, as we define in the section on embeddings 3.3. However, distinguishing unknown meaningful gestures from meaningless hand movements remains an open/unsolved research question which we try to address in this thesis. We try to solve this problem with two approaches discussed in the section on segmentation 3.2.

In deep learning-based recognition models, a latent space is created that maps semantically similar images/gestures to the same label and semantically dissimilar images/gestures to different labels. Our meaningless in-between movements do not have a shared meaning outside of being in-between. Recognising whether a gesture is meaningful or not lies at the heart of open-set recognition [Geng et al. 2021], except that meaningful known gestures are referred to as known known classes (KKCs), meaningful unknown gestures are referred to as unknown known classes (UKCs) and meaningless in-between movements are referred to as known unknown classes (KUCs), sometimes referred to as the background set or the universum set.

[1] As discussed here: `https://stackoverflow.com/questions/17187507/why-use-softmax-as-opposed-to-standard-normalization`

[Dhamija et al. 2018] [Schnyder and Guenther 2021] recently proposed a novel loss function, called the Entropic Open-Set loss, which was proposed to reject unknown classes based on thresholding the output probabilities element-wise. During training time the dataset $\mathcal{D}_{\text{train}}$ contains both meaningful gestures (KKCs) and meaningless gestures (KUCs).

$$J_E(x) = \begin{cases} -\log\left(sm_c(x)\right) & y \in \mathcal{C}_{\text{known}} \text{ of class c} \\ -\frac{1}{|\mathcal{C}_{\text{known}}|} \sum_{c' \in \mathcal{C}_{\text{known}}} \log\left(sm_{c'}(x)\right) & y \in \mathcal{C}_{\text{background}} \end{cases} \tag{2.23}$$

As proven in [Dhamija et al. 2018] J_E is minimised for any $y \in \mathcal{C}_{\text{background}}$ if and only if all Softmax responses are equal. This is only the case, when the input vector to the final Softmax layer, referred to as the deep feature vector F, has equal vector entries itself. We call this subspace $\mathcal{F}_{\text{eq}} = c \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$.

The loss incentivises the model to predict the same probability if the input gesture does not belong to a meaningful class (KUC) by pushing meaningless movements to the subspace $\mathcal{F}_{\text{eq}}$. If the input gesture is a known meaningful gesture, the loss incentivises the model to push all other gestures away from the subspace $\mathcal{F}_{\text{eq}}$. The paper then proceeds by thresholding the output probabilities element-wise to determine whether the input should be rejected or is meaningful.

3

Gesture2Vec

The Gesture2Vec pipeline, see Figure 3.1, is a system for classifying unknown hand gestures in near real-time. First, the video frames containing the unknown gestures are converted into skeletal sequences using a skeletal reconstruction tool supporting hand skeletons like MediaPipe [Lugaresi et al. 2019] or OpenPose [Cao et al. 2018]. Then, Gesture2Vec takes the skeleton sequences and segments them by deciding for each skeletal frame whether or not it contains a meaningful unknown gesture or not. This process is done with two MS-G3D models trained on the greek sign language dataset [Adaloglou et al. 2020], one is trained on the Entropic Open-Set loss, and one is trained on the Cross-Entropy loss with an additional background class. Consecutive skeletal frames with a ”contain meaningful gesture” prediction are then taken together and run through MS-G3D [Liu et al. 2020b] again, extracting embeddings from the latent space of the final two layers. Finally, these unseen embeddings can be used to learn unknown meaningful gestures in an unsupervised way or in a supervised way. We verify the effectiveness of the embeddings by classifying them with supervised machine learning methods introduced in section 3.3.

First, we will introduce the datasets, then we will explain the segmentation process and lastly we explain the Gesture2Vec embeddings. In the next chapter, the effectiveness of these methods is analysed.

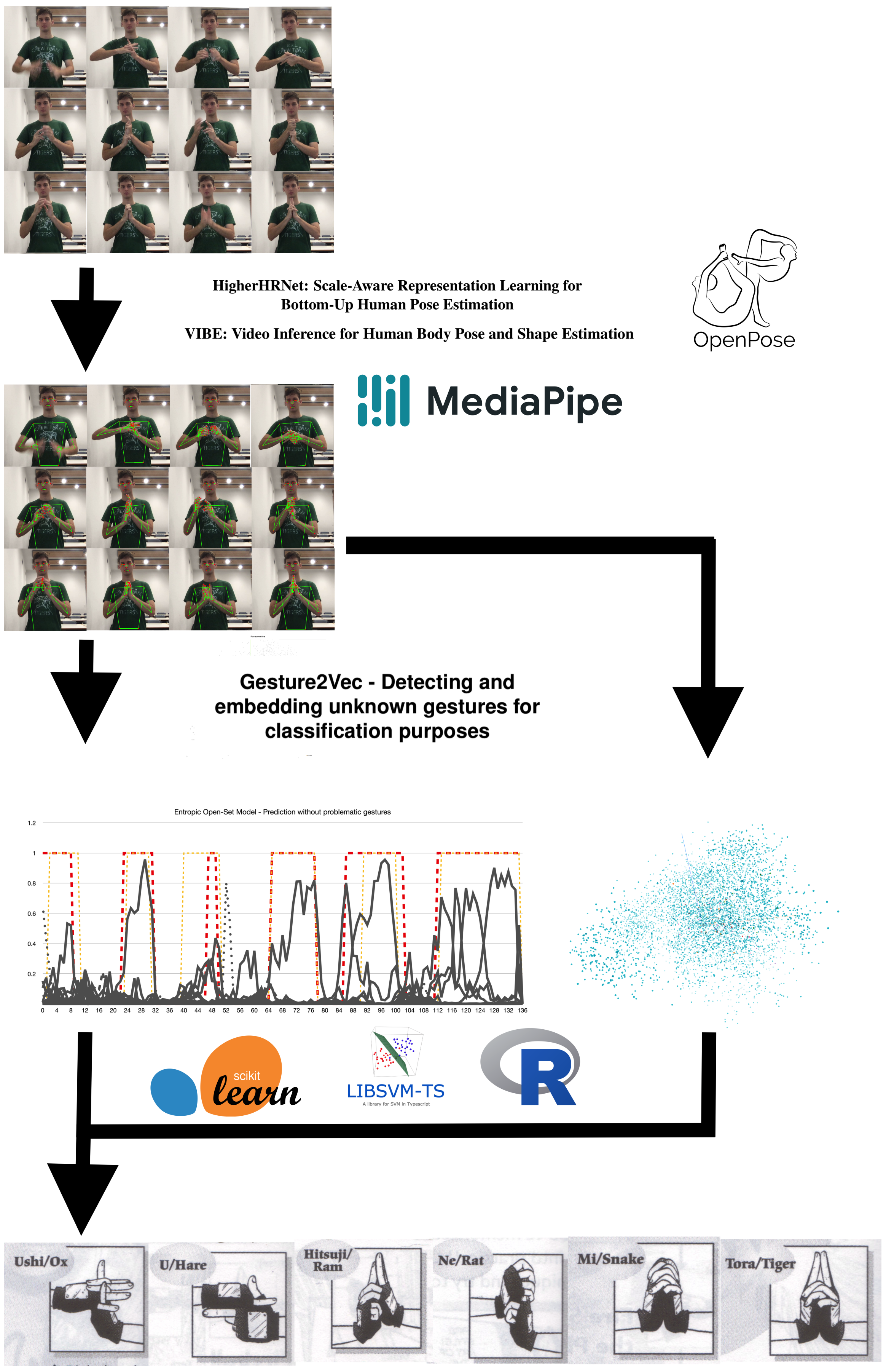

HigherHRNet: Scale-Aware Representation Learning for Bottom-Up Human Pose Estimation
VIBE: Video Inference for Human Body Pose and Shape Estimation
OpenPose
MediaPipe
Gesture2Vec - Detecting and embedding unknown gestures for classification purposes
Entropic Open-Set Model - Prediction without problematic gestures
scikit learn
LIBSVM-TS
A library for SVM in Typescript
R
Ushi/Ox
U/Hare
Hitsuji/ Ram
Ne/Rat
Mi/Snake
Tora/Tiger

Figure 3.1.: *The Gesture2Vec Pipeline, from video sequence, to skeleton sequence, to embeddings and segmentation splits, to the final prediction.*

3.1. Datasets

3.1.1. Greek sign language dataset

The greek sign language dataset [Adaloglou et al. 2020] is a large scale RGB+D dataset which contains 10290 greek sentences. These sentences are represented as a sequence of frames and referred to as the continuous GSL dataset. In addition, [Adaloglou et al. 2020] also provides an isolated sequence where the gestures for each word have been manually segmented.

The initial goal of the dataset was to translate greek sign language videos into greek written language. For this reason, each word, together with etymological annotations, is mapped to an isolated gesture. We proceed to group words that use the same isolated gesture together in the following manner:

Etymological annotations The dataset contains various etymological annotations. For example, the words annotated as ΕΓΩ, ΕΓΩ(1), ΕΓΩ(91), ΕΓΩ910) are grouped together, ie. the etymological annotations discarded.

Noun and verb forms of greek words use the same sign language gesture and are unified. For example, the label ΔΟΥΛΕΙΑstands for the greek noun for work, while the label ΔΟΥΛΕΥΩstands for the greek verb for work.

Euphonic differences Words like ΗΜΈΡΕΣ and ΜΕΡΕΣ are only euphonically different but can be used interchangeably in greek writing and use the same gestures in greek sign language.

Gender of possessive pronouns had to be merged. For example, ΔΙΚΟΣ ΣΟΥ referring to the possessive pronoun yours which is shared with ”ΔΙΚΟ ΣΟΥ” and ”ΔΙΚΉ ΣΟΥ”, which are referring to the same possessive pronoun, but with a different grammatical gender.

Skipped words Some words are skipped in greek sign language. ΑΚΟΥ meaning ”listening” and ΑΚΟΥΩ meaning ”i listen” map to the same gesture.

Reduced form Instead some gestures have a reduced form. ”i listen” happened to have a reduced form which was ΑΚΟΥΩ ΜΕΙΩΝΩ . It has the same gesture (hand to ear) but one touches the gesture with a finger while the other cusps the ear with the hand. Whenever we noted such differences, we labeled them as two seperate gestures.

Miscellaneous words Some words happen to have the same greek sign language representation even though they have a different meaning. ΑΔΕΙΑ meaning ”permission” and ΑΙΤΗΣΗ meaning ”application” is an example of this, see Figure 3.2.

After manually reviewing these gestures, 320 gesture classes were left over (10 more than stated by the greek sign language dataset). 10 additional gesture classes were discarded as we could not map them to a particular word and 22 single gestures were excluded because they could not clearly be assigned to one of the classes. In the published dataset we refer to the discarded labels as ”???”. As manual annotation is error prone, we verified using a cosine similarity test on the embedding vectors that all classes were different from each other. Finally, we extracted all meaningless hand movements in-between meaningful gestures from the continuous dataset.

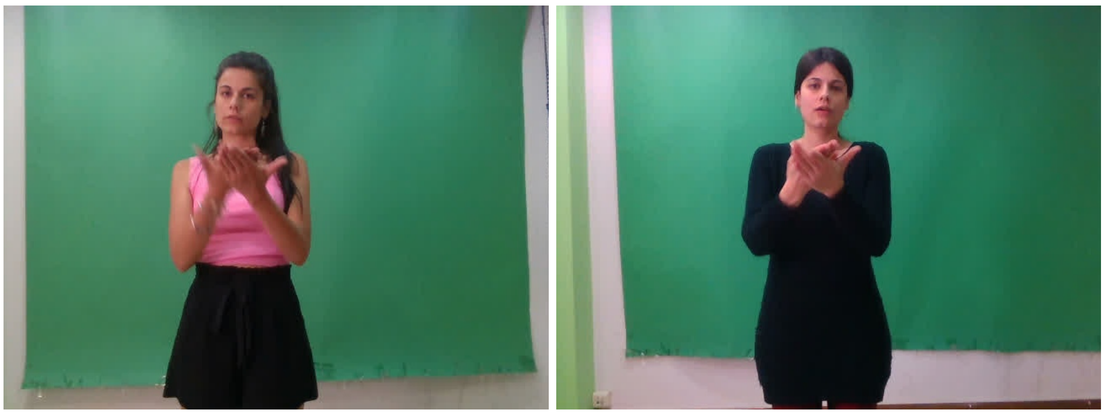


Figure 3.2.: ΑΔΕΙΑ*(permission) and* ΑΙΤΗΣΗ*(application)*

This was done by locating the isolated gestures in the continuous video sequence and extracting the in-between frames. These meaningless movements are labeled as ”background” in the published dataset. Additionally, we removed all sequences in the isolated and continuous dataset where frames were missing.

After reviewing the gestures, we evaluated the two most readily available skeletal gesture recognition frameworks to generate the skeletal representations: OpenPose [Cao et al. 2018] for 2D skeletons and MediaPipe [Lugaresi et al. 2019] for 3D skeletons. OpenPose reconstructs 2D skeletons with confidence values. MediaPipe on the other hand has separate models for 3D body reconstruction and hand skeleton reconstruction. The 3D body model of MediaPipe estimates three joint positions which form a triangle for the hands but no detailed hand skeleton. The hand and body skeletons are merged into one large skeleton by adding edges between the hand joints and the corresponding hand triangle joint of the body reconstruction. The colour mapping in Figure 3.4 represents which hand joints get connected to which body joint. Hand joints of the same colour are each connected by an edge to the corresponding joint on the body of the same colour, see Figure 3.4.

Due to rapid movements of the gestures or occluded fingers, MediaPipe sometimes did not have enough confidence to detect a left or a right hand in the image outside of the triangle (defined below) associated to the body reconstruction, see Figure 3.3. The body recognition never fails on the dataset.

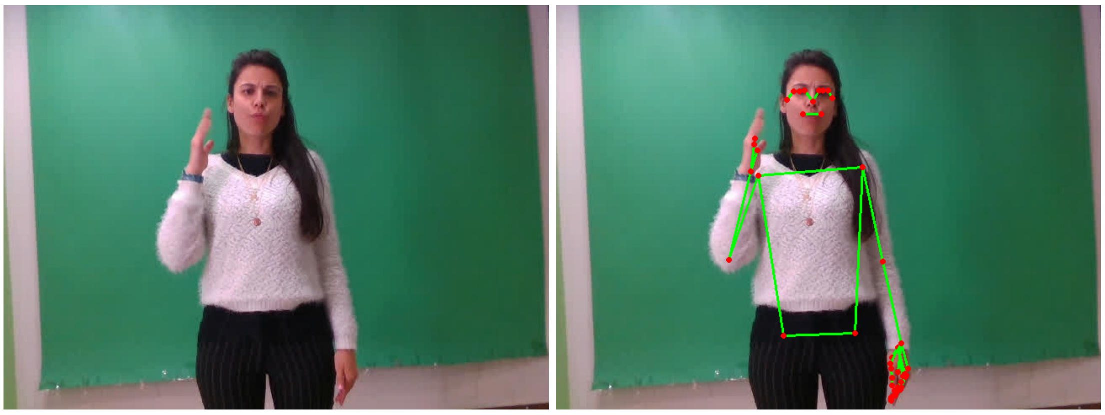


Figure 3.3.: *Sample gesture where the fingers are not found due to occlusion and rapid movement.*

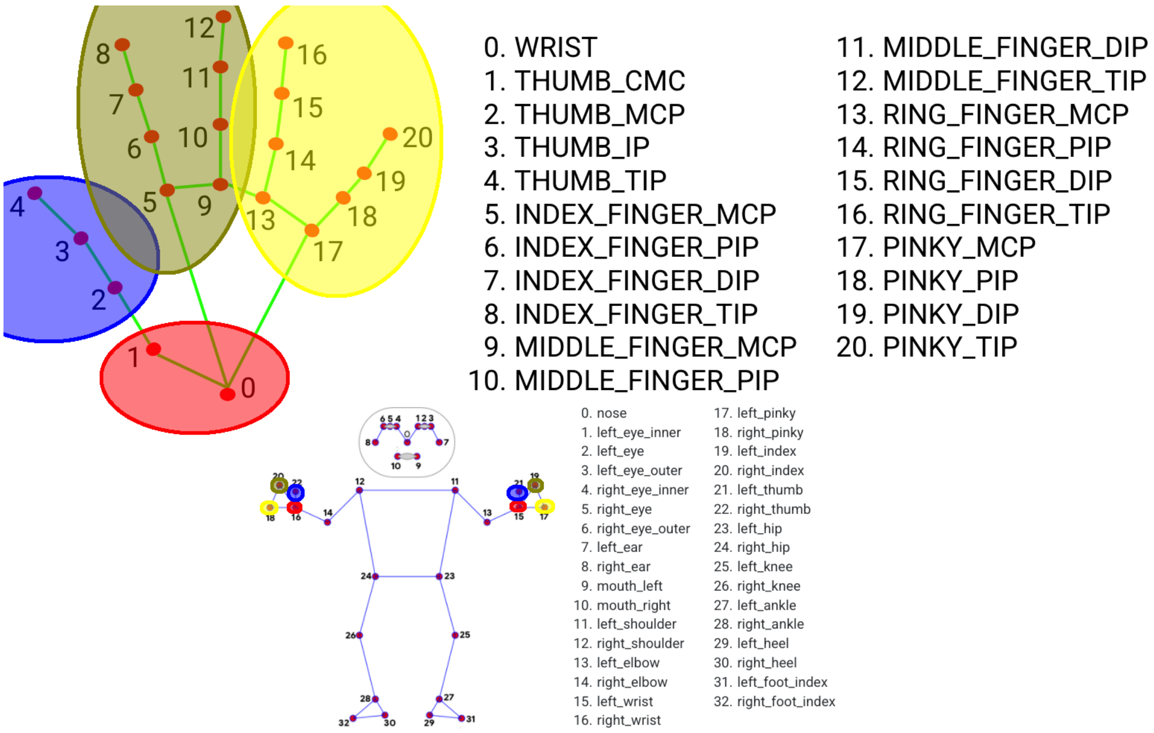


Figure 3.4.: *Missing hand positions take the the corresponding joint position of the body*

3.1.2. Ninja dataset

To get quantitative results for detecting unknown gestures, we created our own isolated ninja dataset based on ninja gestures of 13 classes from [Kishimoto 1999], which are based on Mudra signs, with each at least 13 gesture instances per class, see Table 3.1. For segmenting, we created a separate continuous ninja dataset of two sentences, sentence A using the first half of the gestures in sequence, Bird – Monkey, and sentence B using Ox – Tiger. Each of these two sentences has been repeated 4 times.

Gesture	Number of gestures
Bird	17
Boar	13
Dog	13
Horse	15
Dragon	16
Kagebunshin	15
Monkey	15
Ox	14
Rabbit	14
Ram	15
Rat	15
Snake	14
Tiger	15

Table 3.1.: *Number of isolated gestures per ninja class*

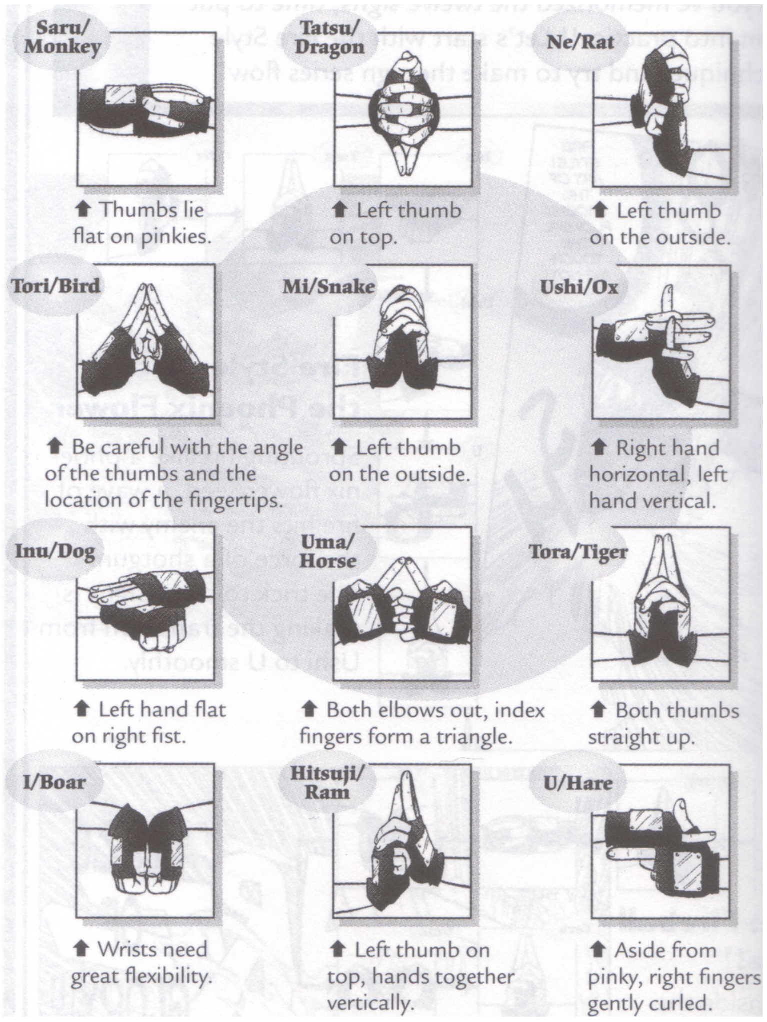

Saru/ Monkey
Thumbs lie flat on pinkies.
Tatsu/ Dragon
Left thumb on top.
Ne/Rat
Left thumb on the outside.
Tori/Bird
Be careful with the angle of the thumbs and the location of the fingertips.
Mi/Snake
Left thumb on the outside.
Ushi/Ox
Right hand horizontal, left hand vertical.
Inu/Dog
Left hand flat on right fist.
Uma/ Horse
Both elbows out, index fingers form a triangle.
Tora/Tiger
Both thumbs straight up.
I/Boar
Wrists need great flexibility.
Hitsuji/ Ram
Left thumb on top, hands together vertically.
U/Hare
Aside from pinky, right fingers gently curled.

Figure 3.5.: *Ninja hand gestures illustrated by [Kishimoto 1999] plus the Kagebunshin sign not listed here*

3.2. Segmenting unknown meaningful gestures

The paper introducing the Entropic Open-Set loss [Dhamija et al. 2018], introduced in the Related Works section 2.4, thresholds the maximum value of the output probabilities to check if an input is meaningful (belongs to a known class) or not, as its primary concern is rejecting unimportant background data. After we did empirical experiments on the final output probabilities of the Entropic Open-Set model trained on the GSL dataset, see Figure 3.6, we found that the output probabilities satisfy the following properties:

1. A meaningful known gesture has a clear significant class.
2. A meaningless background gesture surpasses no significant threshold on any class.
3. A meaningful unknown gesture has one or more significant classes. Combining the features of these significant classes tends to represent the features of the new meaningful unknown gesture, ie. we get a low-dimensional simple embedding.

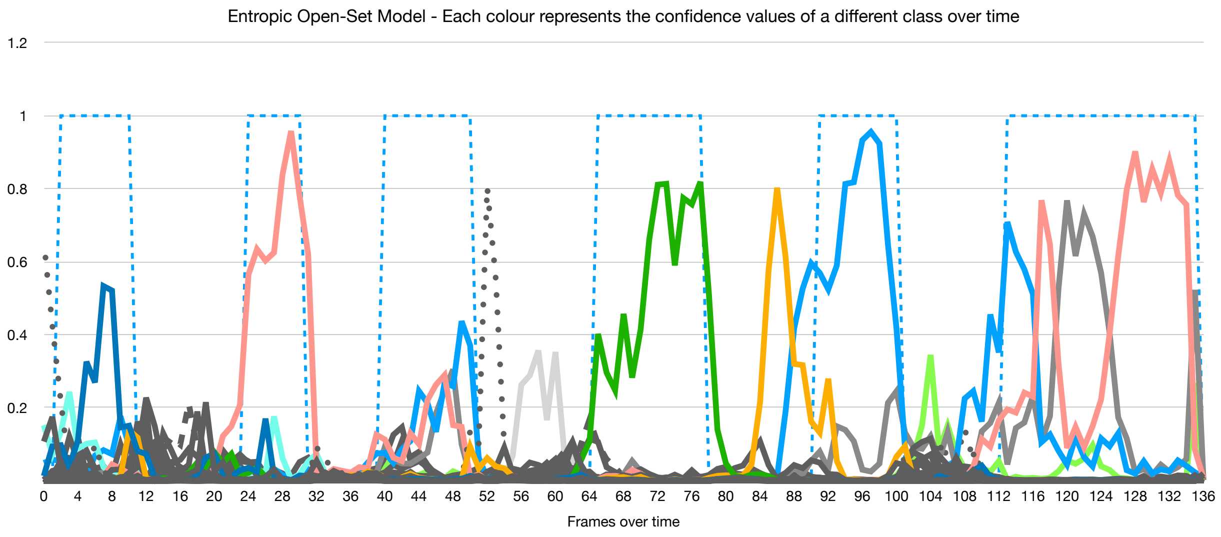


Figure 3.6.: *The confidence values among the 320 gestures reaching significant confidence peaks are coloured differently to signify that unknown gestures can be a linear combination of multiple known gestures. The blue dashed line represents meaningful gestures and the gray dotted line is a background-like gesture.*

3.2.1. Pronounced thresholding

Instead of simply thresholding the Softmax value we use the pronounced gap heuristic [Hiptmair et al. 2016] usually used to find a sufficiently low dimensional representation of the data for principal component analysis . In this way, meaningful unknown gestures, which as we have observed tend to have more than one significant class, are better detected.

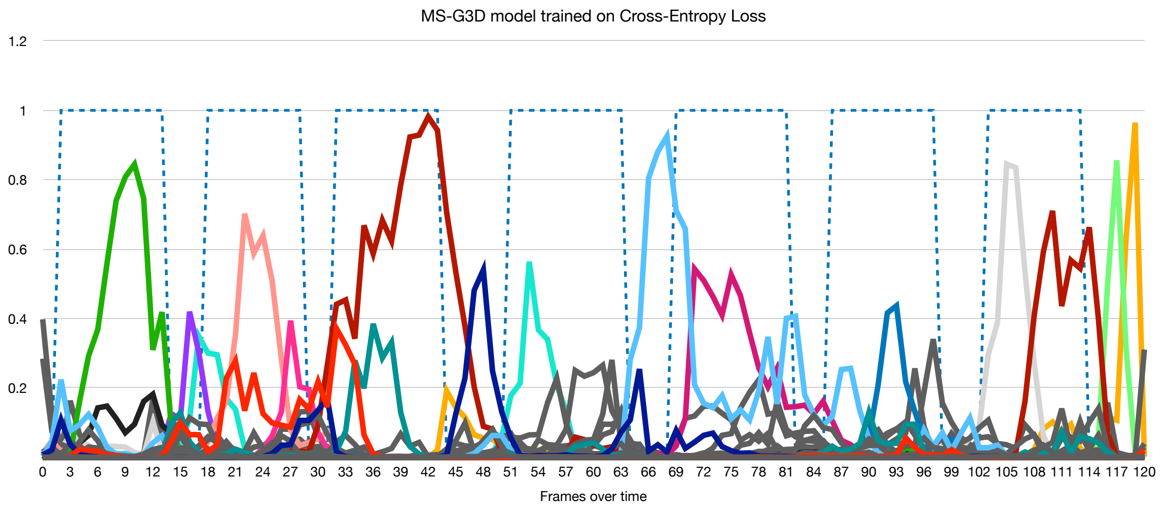
$$k = \min \left\{ q : \sum_{j=1}^{q} \sigma_j^2 \geq (1-\tau) \cdot \sum_{j=1}^{N} \sigma_j^2 \right\} \quad \text{for a threshold } \tau \ll 1 \tag{3.1}$$

Instead of singular values we use the entries of the Softmax probability vector sorted in descending order. This has not been done in previous literature. k is the dimensionality of the gesture with respect to the meaningful gestures. Further, we call the k-dimensional gesture meaningful if the dimensionality is smaller than a specified dimension κ. Intuitively, this is the case when the gesture is only similar to $\leq \kappa$ known gestures.

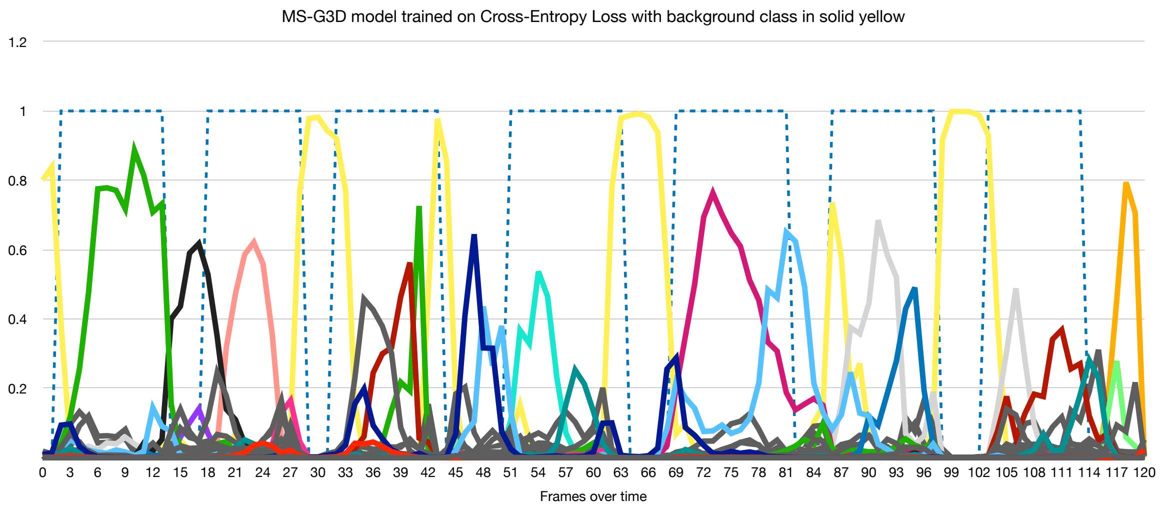
Given a sequence of skeletons containing various meaningful but unknown sign gestures, we create a sliding window for every frame i in the sequence starting at frame i and ending at frame $\tau + i$. Frames for which $\tau + i$ lies outside the video are ignored. We choose τ to be the number of frames of an average unknown gesture[1], in our case 15 frames of a 15fps video, so the sliding window contains exactly one second of the skeletal sequence. Then we check whether each gesture contains a pronounced gap or not and if it does we conclude that frame $i + \lfloor T/2 \rfloor$ is part of a meaningful gesture. The final prediction is a binary vector whose j-th entry predicts whether or not the j-th frame in the sequence contains a meaningful gesture or

[1]The average known gesture meanwhile was 17 frames long due to a few outliers.

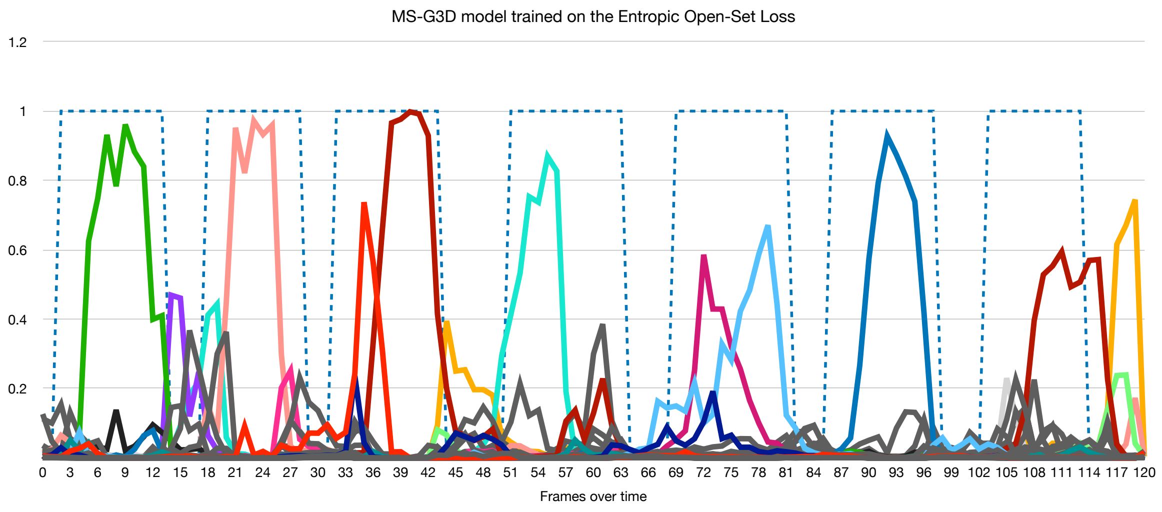
not. Any consecutive 1's in this binary vector are treated as indications of meaningful unknown gestures, while consecutive 0's in this binary vector are treated as indications of gaps between meaningful gestures. The frames are segmented accordingly.



(a) MS-G3D trained on the Cross-Entropy loss



(b) MS-G3D trained on the Cross-Entropy loss with an additional background class, where the background class is marked in solid yellow



(c) MS-G3D trained on the Entropic Open-Set loss

Figure 3.7.: *Sentence A1 filled with 7 unknown meaningful gestures from our continuous ninja dataset. At frame x we get the final probabilities from the respective models for the gesture starting at frame x and lapping over to frame $x + k$, in this case $k = 15$. The dotted blue lines are ground-truth annotations of where the unknown meaningful gestures start and end. The solid yellow line represents the background class. All other solid colours represent other classes.*

3.2.2. Denoising

Oftentimes the binary vector from the pronounced thresholding step is noisy, meaning a gesture is wrongly detected for a single frame or a gap is detected for a single frame. We can safely assume that gestures and gaps between gestures happen for longer than one frame, so we can apply the following denoising procedure to smooth the model's predictions over time.

We define a minimal gesture length gest_{min} and a minimal gap length gap_{min} (in frames). These are removed by first removing the gestures of length 1, then removing gaps of length 1. Next, gestures of length 2 are removed and then gaps of length 2 are removed and so on until the respective minimal lengths are reached.

```
gestmin = 4
gapmin = 3

for i in max(gestmin, gapmin):
   for pred in predictions:
      if pred == 1 and len(neighborhood(pred)) <= min(i,gestmin):
         pred = 0

   for pred in predictions:
      if pred == 0 and len(neighborhood(pred)) <= min(i,gapmin):
         pred = 1
```

Figure 3.8.: *Denoising pseudocode*

3.2.3. Background-like gestures

There were two gestures that kept appearing at positions where there should be no meaningful gestures only meaningless movement. The gesture ΓΕΝΝΩ 3.9, which is a gesture where the hands are moved vertically downwards, something you constantly do between two meaningful gestures. Furthermore, ΕΝΤΑΞΕΙ 3.10 is a gesture in which your hands are pulled apart horizontally, as though you just finished a gesture.

Ignoring these two gestures leads to slightly better segmentation results. For a comparison, see Figure 3.11. We can remove a gesture either before applying the softmax by setting it to $-\infty$ or after, by taking the log of all the softmax values, removing the gesture by setting its entry to $-\infty$ and recomputing the softmax values. We proceed to prove that this is equivalent.

$$S = \operatorname{softmax}\left(\begin{bmatrix} c_1 \\ \vdots \\ c_{i-1} \\ -\infty \\ c_{i+1} \\ \vdots \\ c_N \end{bmatrix}\right) \overset{!}{=} \operatorname{softmax}\left(\log\left(\operatorname{softmax}\begin{bmatrix} c_1 \\ \vdots \\ c_{i-1} \\ c_i \\ c_{i+1} \\ \vdots \\ c_N \end{bmatrix}\right) + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -\infty \\ 0 \\ \vdots \\ 0 \end{bmatrix}\right) = S' \tag{3.2}$$

We prove this case-wise over the vector entries k.

Case $k \neq i$:

$$\begin{aligned}
S'_k &:= \frac{e^{\log\left(\frac{e^{c_k}}{\sum_q e^{c_q}}\right)} + 0}{\sum_j e^{\log\left(\frac{e^{c_j}}{\sum_q e^{c_q}}\right) + 1_{[j=i]}\cdot(-\infty)}} && \text{(3.3)} \\
&= \frac{e^{\log\left(\frac{e^{c_k}}{\sum_q e^{c_q}}\right)} + 0}{e^{\log\left(\frac{e^{c_i}}{\sum_q e^{c_q}}\right) - \infty} + \sum_{j\neq i} e^{\log\left(\frac{e^{c_j}}{\sum_q e^{c_q}}\right) + 0}} && \text{(3.4)} \\
&= \frac{e^{c_k - \log\left(\sum_q e^{c_q}\right)}}{\sum_{j\neq i} e^{c_j - \log\left(\sum_q e^{c_q}\right)}} && \text{(3.5)} \\
&= \frac{\frac{e^{c_k}}{e^{\log\left(\sum_q e^{c_q}\right)}}}{\frac{\sum_{j\neq i} e^{c_j}}{e^{\log\left(\sum_q e^{c_q}\right)}}} && \text{(3.6)} \\
&= \frac{e^{c_k}}{\sum_{j\neq i} e^{c_j}} && \text{(3.7)} \\
&= \frac{e^{c_k}}{\sum_j e^{1_{[i\neq j]}\cdot c_j}} \qquad := S_k && \text{(3.8)}
\end{aligned}$$

Else:

$$
\begin{aligned}
S_i' &:= \frac{e^{\log\left(\frac{e^{-\infty}}{\sum_q e^{c_q}}\right)} + 0}{\sum_j e^{\log\left(\frac{e^{c_j}}{\sum_q e^{c_q}}\right)+1_{[j=i]}\cdot(-\infty)}} && (3.9) \\
&= \frac{e^{\log(0)}}{\sum_j e^{\log\left(\frac{e^{c_j}}{\sum_q e^{c_q}}\right)+1_{[j=i]}\cdot(-\infty)}} && (3.10) \\
&= \frac{e^{-\infty}}{\sum_j e^{\log\left(\frac{e^{c_j}}{\sum_q e^{c_q}}\right)+1_{[j=i]}\cdot(-\infty)}} && (3.11) \\
&= 0 && (3.12) \\
&= \frac{e^{-\infty}}{\sum_j e^{1_{[i\neq j]}\cdot c_j}} \qquad =: S_i && (3.13)
\end{aligned}
$$

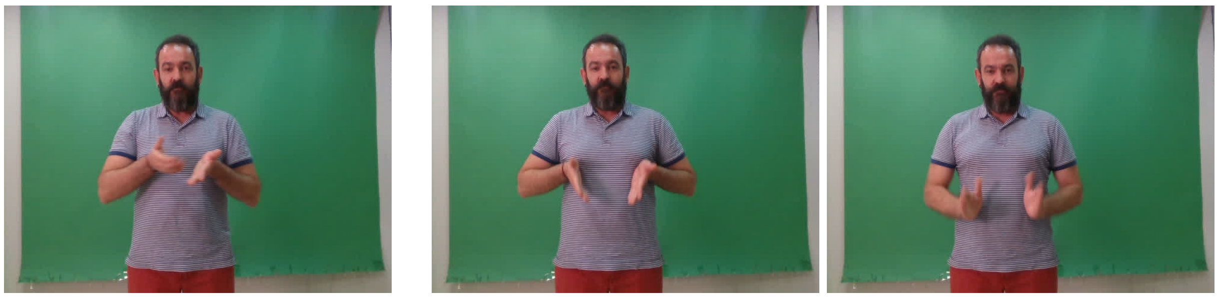

(a) Frame 0 (b) Frame 8 (c) Frame 15

Figure 3.9.: ΓΕΝΝΩ *moving hands downward and then reorienting them gesture.*

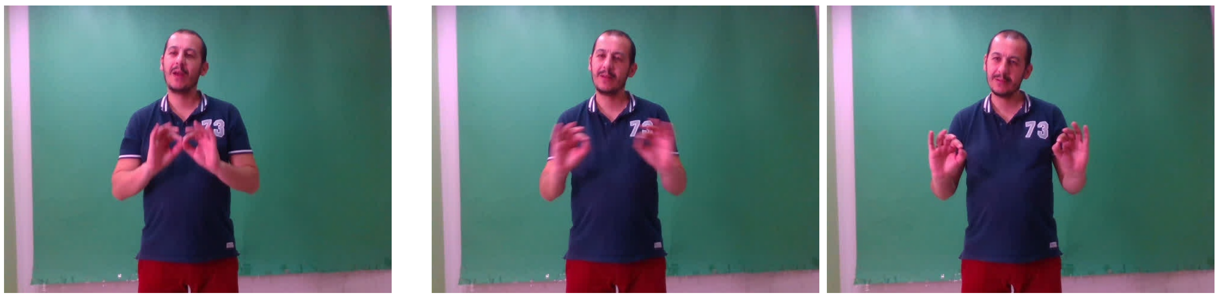
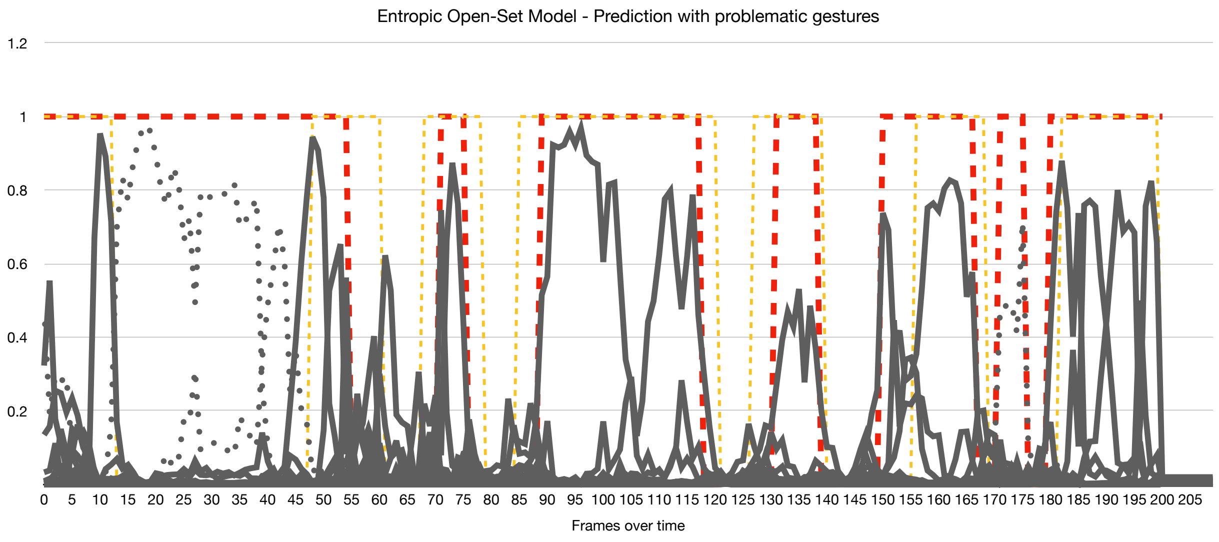
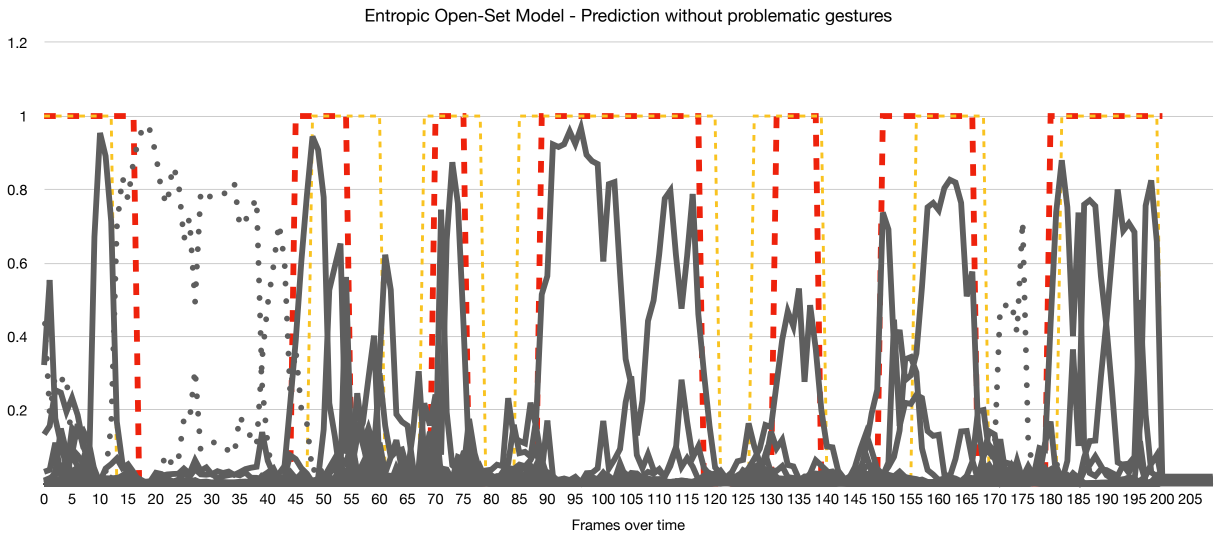

(a) Frame 0 (b) Frame 5 (c) Frame 11

Figure 3.10.: ΕΝΤΑΞΕΙ *gesture pulling apart the hands horizontally.*



(a) Prediction with the background-like gestures



(b) Prediction without the background-like gestures

Figure 3.11.: *Removing both* ΓΕΝΝΩ *and* ΕΝΤΑΞΕΙ *(the dotted gray lines) has the following effect on sentence B1 of the continuous ninja dataset (both denoised). Red is the prediction.*

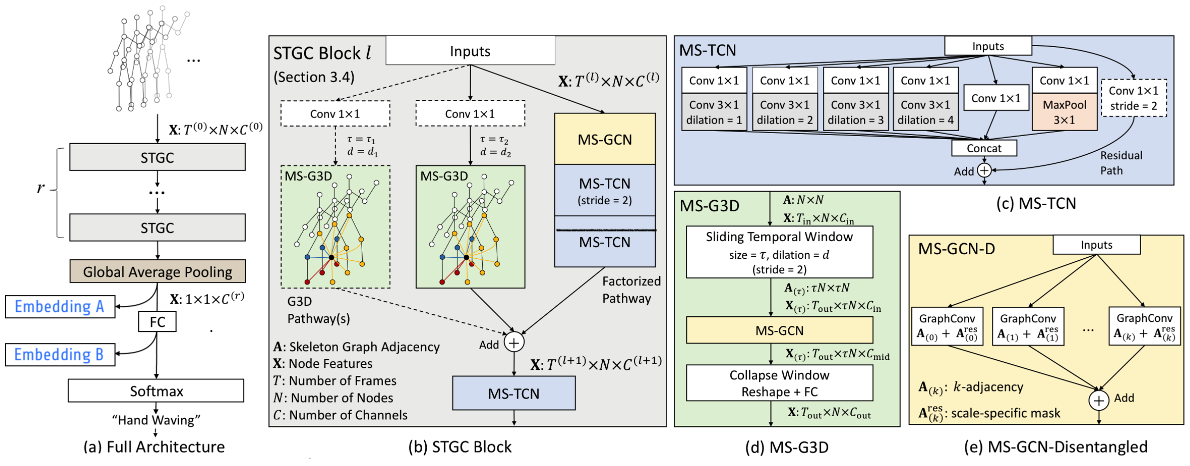
3.3. Embeddings

We evaluate three different embeddings extracted from the MS-G3D model trained on the greek sign language dataset.

Using the standard settings, the size after the global pooling layer of the MS-G3D model has size $\text{batchsize} \times 384 \times 36 \times 75 \times 1$ (batch size $\times$ last channel size $\times$ temporal dimensions $\times$ joint count $\times$ number of skeletons). Then a mean pooling layer is applied on the spatial & temporal dimensions and the number of people, shrinking the size of the vector to $\text{batchsize} \times 384$. This is where we extract **Embedding A**, denoted in the Figure 3.12. Then a single perceptron forward layer of input dimension 384 and output dimension 350 (number of gestures) is applied to the network, after which a standard unit Softmax function is applied to get the gesture probabilities. **Embedding B** is extracted right before applying the Softmax on the last layer, see Figure 3.12. These embeddings of size 384 and 350 per sample are then concatenated into:

$$\textbf{Embedding C} = \Big[\textbf{Embedding A} \quad | \quad \textbf{Embedding B}\Big] \tag{3.14}$$

The embeddings have been uploaded to the Tensorflow Embedding Projector for further study, see Figure 3.13 and footnotes [2,3]



$\mathbf{X}: T^{(0)}\times N\times C^{(0)}$
STGC
r
STGC
Global Average Pooling
Embedding A
$\mathbf{X}: 1\times 1\times C^{(r)}$
FC
Embedding B
Softmax
"Hand Waving"
(a) Full Architecture
STGC Block l
(Section 3.4)
Inputs
$\mathbf{X}: T^{(l)}\times N\times C^{(l)}$
Conv 1×1
Conv 1×1
$\tau=\tau_1$ $d=d_1$
$\tau=\tau_2$ $d=d_2$
MS-G3D
MS-G3D
MS-GCN
MS-TCN (stride = 2)
MS-TCN
Factorized Pathway
G3D Pathway(s)
Add
$\mathbf{X}: T^{(l+1)}\times N\times C^{(l+1)}$
MS-TCN
A: Skeleton Graph Adjacency
X: Node Features
T: Number of Frames
N: Number of Nodes
C: Number of Channels
(b) STGC Block
MS-TCN
Inputs
Conv 1×1
Conv 3×1 dilation = 1
Conv 3×1 dilation = 2
Conv 3×1 dilation = 3
Conv 3×1 dilation = 4
Conv 1×1
MaxPool 3×1
Conv 1×1 stride = 2
Concat
Add
Residual Path
(c) MS-TCN
MS-G3D
$\mathbf{A}: N\times N$
$\mathbf{X}: T_{\text{in}}\times N\times C_{\text{in}}$
Sliding Temporal Window size = τ, dilation = d (stride = 2)
$\mathbf{A}_{(\tau)}: \tau N\times\tau N$
$\mathbf{X}_{(\tau)}: T_{\text{out}}\times\tau N\times C_{\text{in}}$
MS-GCN
$\mathbf{X}_{(\tau)}: T_{\text{out}}\times\tau N\times C_{\text{mid}}$
Collapse Window Reshape + FC
$\mathbf{X}: T_{\text{out}}\times N\times C_{\text{out}}$
(d) MS-G3D
MS-GCN-D
Inputs
GraphConv $\mathbf{A}_{(0)}+\mathbf{A}^{\text{res}}_{(0)}$
GraphConv $\mathbf{A}_{(1)}+\mathbf{A}^{\text{res}}_{(1)}$
GraphConv $\mathbf{A}_{(k)}+\mathbf{A}^{\text{res}}_{(k)}$
$\mathbf{A}_{(k)}$: k-adjacency
$\mathbf{A}^{\text{res}}_{(k)}$: scale-specific mask
Add
(e) MS-GCN-Disentangled

***Figure 3.12.:** MS-G3D architecture with its extracted Embeddings*

3.3.1. Validate the effectiveness of the embeddings

To validate the effectiveness of the embeddings, we use various supervised learning techniques which do not assume a simple structure on the dimensions, as the latent space feature dimensions with complex meaning. We are interested in the supervised scenario, as we want users to define gestures at run-time and be able to use them immediately. The performance of the supervised methods should serve as an upper-bound to unsupervised clustering performance. We use the following supervised techniques:

KNN(x) is the k-nearest neighbour algorithm applied on x neighbours.

MLP(x) is a multilayer perceptron model with 1 hidden layer of size x to classify the gestures.

[2]Embedding A, isolated ninja gestures only, standard loss: `https://projector.tensorflow.org/?config=https://gist.githubusercontent.com/bvoq/5732374047d963bfb344ee02d8772a4c/raw/c39f0396fbca38b3b10aa22e8d74e6e721181f34/projector`

[3]Embedding A, with isolated ninja gestures and the entire validation set of isolated GSL (signer 7), standard loss: `https://projector.tensorflow.org/?config=https://gist.githubusercontent.com/bvoq/7f9f61df751f1ebf369ed8eff2c2bc01/raw/1808932deafb0839efff9af99a855a238cdf4284/project`

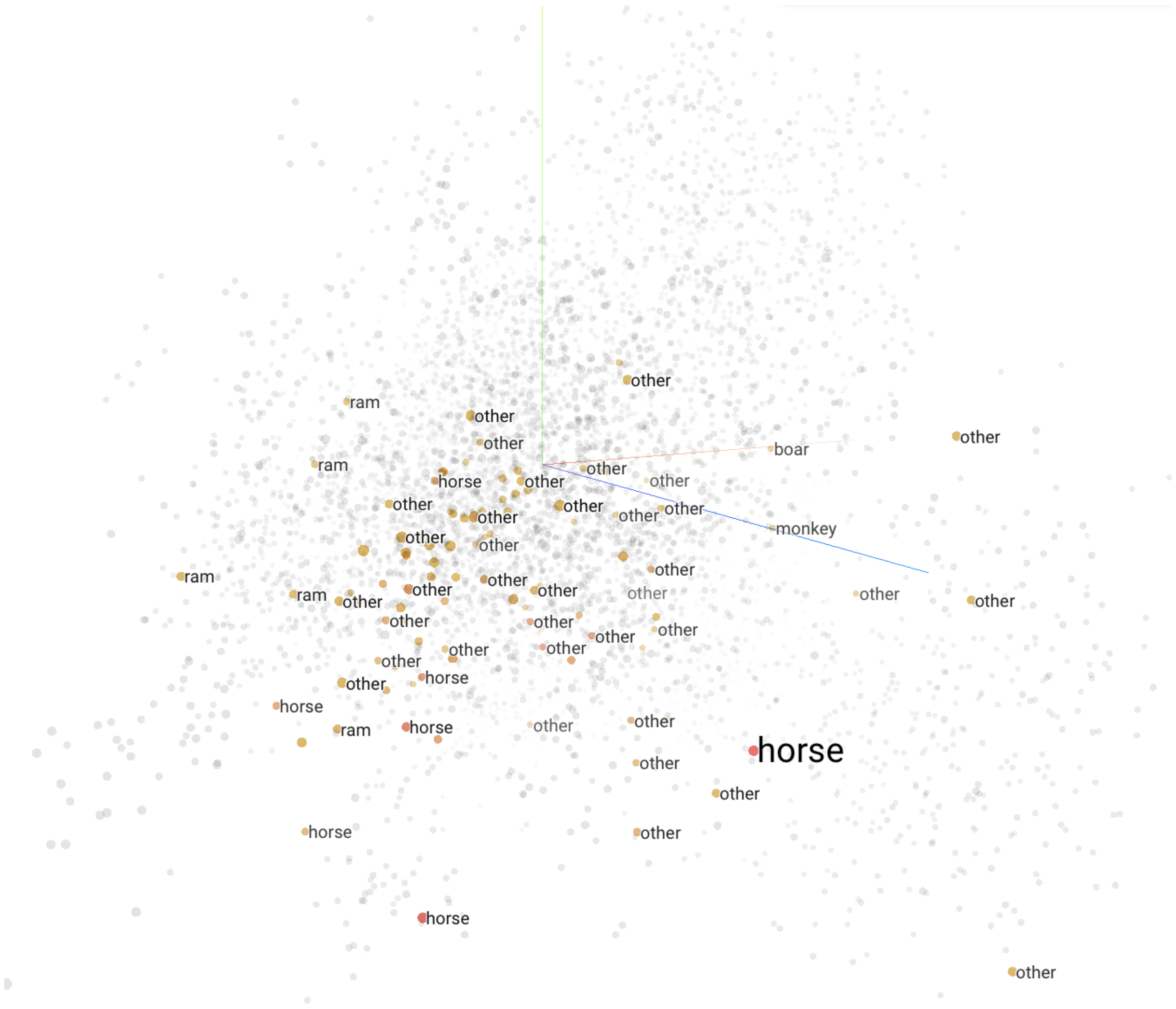


Figure 3.13.: *Embeddings visualised with the Tensorflow projector.*

RandomForest is a random forest classifier with 100 estimators using the Gini criteria to measure a split's quality.

SVC is a degree 3 multi-class support vector machine with the one-vs-one scheme. A radial basis kernel is used with $\gamma = \frac{1}{|C| * \text{var}(X)}$, where X is the embedding matrix, with a squared l-2 regulariser of 1.0.

GaussianNB is a Gaussian Naive Bayes classifier with no priors.

Bagging[y] A square bracket indicates that aggregation of size y is applied to the model. Aggregation is a technique to reduce variance among multiple instances of the same model by taking the plurality of the predictions of these y models. For example, MLP(100)[10] aggregates the predictions of 10 MLPs with a hidden layer of size 100.

4

Results

4.1. Performance of MS-G3D on the Greek Sign Language Dataset

The greek sign language dataset consists of 320 classes and 7 signers, each repeating the same gestures/sentences 5 times. We validate all our models on one signer while training the model on the other 6, so our performance scores are cross-signer validated scores.

We use three different models trained in the following manner:

$\mathcal{M}_{\textbf{standard}}$**:** is the MS-G3D model trained on the meaningful greek sign language gestures with the Cross-Entropy loss.

$\mathcal{M}_{\textbf{background-as-class}}$**:** is the MS-G3D model trained on the meaningful greek sign language gestures and on the meaningless in-between movements as a separate background class with the Cross-Entropy loss.

$\mathcal{M}_{\textbf{entropic-openset}}$**:** is the MS-G3D model trained on the meaningful greek sign language gestures and on the meaningless in-between movements with the Entropic Open-Set loss.

For training, all the MS-G3D models used 350 output classes, 320 of which were trained gestures. Class 321 was used for training a background class in model $\mathcal{M}_{\text{background-as-class}}$. The other classes were untrained auxiliary classes leaving spare room for labelling unknown gestures when extracting embeddings. We use NVIDIA Apex 16-bit floating point numbers during training and as parameters. $\mathcal{A}_{\text{res}}$ was initialised uniformly at random with parameters between $[-1\text{e-}6, 1\text{e-}6]$ and $\mathcal{W}^{(p)}$ was initialised using He's initialisation [He et al. 2015] with scaling paramater $a = \sqrt{5}$ [1]. We train the models using stochastic gradient descent using a base learning rate of 0.05 with 65 epochs, Nesterov momentum of 0.9 to dampen oscillations and a $l2$-regularisation/weight decay parameter of 0.0003. The learning rate is halved at 45 epochs and again at 55 epochs. The batch size is 16 and the forward batch size is 8 for the Cross-Entropy loss-based models $\mathcal{M}_{\text{standard}}$ and $\mathcal{M}_{\text{background-as-class}}$ and for the Entropic Open-Set loss-based model $\mathcal{M}_{\text{entropic-openset}}$ the batch size is 8 with forward batch size 4.

Using the standard settings, the three STGCN modules have output channels sized $\{96, 192, 384\}$ with window size $\tau = 8$ and dilation stride $d = 1$ for the first module and $d = 2$ for the second and third module. The starting frame size is $T = 144$ (padded with zeros), which is also used for the first module, after which it is halved to 72 and finally 36 in subsequent layers.

On the greek dataset transformed with MediaPipe, we achieve a top-1 classification score of roughly 87% while OpenPose achieves a top-1 classification score of roughly 66%, see Table 4.1. All other scores are evaluated on the MediaPipe trained models. Due to GPU memory limitations, we trained the Entropic Open-Set model with batch size 8 and forward batch size 4, while the models with a Cross-entropy loss were trained with batch size 16 and forward batch size 8. Other than that, all the models have been trained with the same hyperparameters and number of epochs.

Model	Skeleton data	Cross-signer validation Top-1	Cross-signer validation Top-5	Training Time
$\mathcal{M}_{\text{standard}}$	OpenPose	66.13%	87.81%	2576 min
$\mathcal{M}_{\text{standard}}$	MediaPipe	87.08%	96.23%	2917 min
$\mathcal{M}_{\text{entropic-openset}}$	MediaPipe	87.67%	96.32%	6907 min
$\mathcal{M}_{\text{background-as-class}}$	MediaPipe	86.01%	94.14 %	4824 min

Table 4.1.: *Performance of MS-G3D on the Greek Sign Language dataset [Adaloglou et al. 2020] after converting it with MediaPipe [Lugaresi et al. 2019] or OpenPose [Cao et al. 2018] into skeletal sequences on the validation signer. The training times are different due to learning background movements (almost doubling the dataset), skeleton joint count and batch sizes.*

4.2. Segmentation

Segmentation is done with the models $\mathcal{M}_{\text{background-as-class}}$ by thresholding the background class accuracy by τ and with $\mathcal{M}_{\text{entropic-openset}}$ with threshold parameter τ and maximum meaningful dimensionality κ as described in section 3.2.

The continuous ninja dataset, on which the segmentation is evaluated, is limited to 2 sentences containing the 13 unknown meaningful gestures from the ninja dataset with 4 repetitions each, totalling $4 \cdot 7 + 4 \cdot 6 = 52$ gestures. Because it is hard to determine the start and end positions of gestures precisely, we instead count the number of correct overlaps of detected gestures/gaps

[1] `https://github.com/pytorch/pytorch/issues/15314`

with the ground-truth gestures/gaps. An unaccounted gap or gesture in either the predictions or the ground-truth label increases the error. The relative error is measured by dividing the error by all gaps and gestures in the predicted and ground-truth labels. An implementation of the precise error can be seen in the appendix, Figure A.1.

We grid-search τ (the threshold) and κ (the maximum meaningful dimensionality) based on this error for a given gest_{min} and gap_{min}. By increasing the threshold τ, the length of the gestures becomes larger, but the length of the gaps become smaller. If the threshold τ is increased by a lot, the segmentation method detects more gestures but fewer gaps. Similarly, if the threshold τ is decreased by a lot, the segmentation method detects more gaps but fewer gestures. We present the optimal values of τ, κ, which minimise the error-metric for each of the segmentation methods and achieve a relative accuracy rate of 95% relative accuracy based on this error-metric as seen in Table 4.2.

Note that the model $\mathcal{M}_{\text{background-as-class}}$ trained on the Cross-Entropy loss has background class activations that happen only for a frame or two. For this reason, we get better results if we keep gap_{min} low for this model as shown by Table 4.2. The methods can be compared visually in Figures A.3,A.4, A.5, A.6, A.7, A.8, A.9 and A.10.

Table 4.2.: *Segmentation performances, EOSL=Entropic Open-Set Loss, CEL=Cross-Entropy Loss*

Segmentation Method	κ	τ	gest_{min}	gap_{min}	missing gestures	missing gaps	Relative error
EOSL based segmentation	8	0.007	3	3	2	4	0.053
EOSL based segmentation wo. background-like gestures	16	0.0055	4	4	1	4	0.052
CEL with background class based segmentation	-	0.071	4	4	0	6	0.065
CEL with background class based segmentation	-	0.071	2	6	1	4	0.040

4.3. Recognition of unknown gestures

To test the effectiveness of our embedding, we classify the unknown meaningful gestures using various supervised learning techniques. We divide our results based on the number of samples needed to get an accurate result. The model settings are explained in section 3.3.

4.3.1. One-shot scenario

In the one-shot scenario, we have one embedding of a gesture from each class as our training set. To classify a new gesture, we find its *nearest neighbour* in the embedding space.

KNN(1) - Only ninja dataset: In this scenario, we have precisely one embedding corresponding to a gesture for each of the 13 ninja classes picked at random as a training set. Then we

Model	Algorithm	Embedding A	Embedding B	Embedding C
$\mathcal{M}_{\text{standard}}$	KNN(1)	**0.61925**	0.59998	0.600454
$\mathcal{M}_{\text{entropic-openset}}$	KNN(1)	0.59604	0.59042	0.58992
$\mathcal{M}_{\text{background-as-class}}$	KNN(1)	0.59778	0.58664	0.58948

Table 4.3.: *1-shot performance on the ninja dataset. 13000 models were trained per entry and their performance averaged. Embedding A, extracted from model $\mathcal{M}_{\text{standard}}$, has the highest classification accuracy of* 61.925%.

Model	Algorithm	Embedding A	Embedding B	Embedding C
$\mathcal{M}_{\text{standard}}$	KNN(1)	**0.5775**	0.5436	0.5520
$\mathcal{M}_{\text{entropic-openset}}$	KNN(1)	0.55415	0.54462	0.54585
$\mathcal{M}_{\text{background-as-class}}$	KNN(1)	0.53992	0.51631	0.51369

Table 4.4.: *1-shot performance on the ninja+greek dataset. 13000 models were trained per entry and their performance averaged. Embedding A, extracted from model $\mathcal{M}_{\text{standard}}$, has the highest classification accuracy of* 57.775%.

classify a gesture, which is not used in the training set, chosen at random from the 13 ninja classes. This training and classification procedure is repeated 13000 times, with 1000 random classification samples from each class. We get a final accuracy of around $58 - 62\%$. See the results in Table 4.3.

KNN(1) - Greek and ninja dataset: In this scenario, we have precisely one embedding corresponding to a gesture for each of the 13 ninja classes and each of the 320 greek sign language classes picked at random as a training set. Then we classify a gesture, which is not used in the training set, chosen at random from the 13 ninja classes. This training and classification procedure is repeated 13000 times, with 1000 random classification samples from each class. We get a final accuracy of around $51 - 58\%$. See the results in Table 4.4. Embedding A, extracted from model $\mathcal{M}_{\text{standard}}$, has the highest classification accuracy of 57.775%.

Algorithm	Embedding A	Embedding B	Embedding C
KNN(1)	0.7956	0.7682	0.7727
KNN(2)	0.7822	0.7508	0.7552
KNN(3)	0.7576	0.7260	0.7287
KNN(4)	0.7578	0.7319	0.7464
KNN(5)	0.7361	0.7185	0.7168
KNN(6)	0.7296	0.7064	0.7056
SVC[10]	0.8002	0.7997	0.8144
RandomForest[10]	0.7987	0.7845	0.8044
MLP(100)[10]	**0.8753**	0.8368	0.8438

Table 4.5.: *4-shot performance on embeddings extracted from model $\mathcal{M}_{\text{standard}}$. 13000 models were trained per entry and their performance averaged. The best performance was achieved by Embedding A trained on MLP(100)[10] with 87.53% classification accuracy.*

4.3.2. 4-shot scenario

In the 4-shot scenario, we choose a set of 4 embeddings per class as our training set. Then an embedding is chosen uniformly at random from a given class to be predicted. To classify a new embedding we use machine learning techniques such as K-Nearest-Neighbour, Multilayer Perceptron with a hidden layer of size 100, Support Vector Classification one-vs-one, Random-Forests, Gaussian Naive Bayes as discussed in 3.3. Each method is trained 13000 times on different 4 randomly chosen embeddings per class, creating 13000 models. 1000 embeddings from each class are picked at random and evaluated by one of the 13000 models, in such a way, that the embedding to be predicted does not lie in the training set.

Embedding A, extracted from model $\mathcal{M}_{\text{standard}}$, trained on 10 aggregated neural networks with a single hidden layer size of 100, has the highest classification accuracy of 87.53%. See Table 4.5, 4.6 and 4.7 for the results.

4.3.3. Entire scenario

Here we use leave-one-out cross validation (LOOCV) to obtain our results on the entire ninja dataset to prevent overfitting, see Table 4.8, 4.9 and 4.10. Embedding B, extracted from model $\mathcal{M}_{\text{standard}}$, trained on 10 aggregated random forests, has the highest classification accuracy of 94.21%.

Algorithm	Embedding A	Embedding B	Embedding C
KNN(1)	0.7759	0.7667	0.7745
KNN(2)	0.7481	0.7448	0.7367
KNN(3)	0.7418	0.7352	0.7385
KNN(4)	0.7435	0.7238	0.7348
KNN(5)	0.7171	0.7065	0.7184
KNN(6)	0.7057	0.6975	0.6993
SVC[10]	0.7716	0.8021	0.8117
RandomForest[10]	0.7847	0.7654	0.7889
MLP(100)[10]	**0.8414**	0.8286	0.8376

Table 4.6.: *4-shot performance on embeddings extracted from model $\mathcal{M}_{\text{entropic-openset}}$. 13000 models were trained per entry and their performance averaged. The best performance was achieved by Embedding A trained on MLP(100)[10] with* 84.14% *classification accuracy.*

Algorithm	Embedding A	Embedding B	Embedding C
KNN(1)	0.7655	0.7538	0.7477
KNN(2)	0.7438	0.7348	0.7344
KNN(3)	0.7372	0.7232	0.7276
KNN(4)	0.7303	0.7258	0.7288
KNN(5)	0.7094	0.7087	0.7154
KNN(6)	0.6920	0.6926	0.6914
SVC[10]	0.7425	0.7366	0.7858
RandomForest[10]	0.7692	0.7530	0.7764
MLP(100)[10]	**0.8368**	0.7987	0.8161

Table 4.7.: *4-shot performance on embeddings extracted from model $\mathcal{M}_{\text{background-as-class}}$. 13000 models were trained per entry and their performance averaged. The best performance was achieved by Embedding A trained on MLP(100)[10] with* 83.68% *classification accuracy.*

Algorithm	Embedding A	Embedding B	Embedding C
KNN(1)	0.8684	0.8316	0.8316
KNN(2)	0.8684	0.8368	0.8421
KNN(3)	0.8895	0.8579	0.8684
KNN(4)	0.8947	0.8526	0.8579
KNN(5)	0.8579	0.8579	0.8632
KNN(6)	0.8789	0.8737	0.8737
KNN(7)	0.8579	0.8211	0.8263
KNN(8)	0.8579	0.8421	0.8421
KNN(9)	0.8474	0.8316	0.8368
KNN(10)	0.8737	0.8474	0.8474
KNN(11)	0.8474	0.8105	0.8158
KNN(12)	0.8474	0.8105	0.8105
MLP(100)[10]	0.9158	0.9368	0.9158
RandomForest[10]	0.9105	**0.9421**	0.9263
SVC[10]	0.8895	0.9053	0.8895
GaussianNB[10]	0.8789	0.8263	0.8579

Table 4.8.: *LOOCV on embeddings extracted from the model $\mathcal{M}_{standard}$. Embedding B trained on RandomForest[10] achieved the highest accuracy of 94.21%.*

Algorithm	Embedding A	Embedding B	Embedding C
KNN(1)	0.8632	0.8579	0.8579
KNN(2)	0.8526	0.8421	0.8474
KNN(3)	0.8789	0.8526	0.8526
KNN(4)	0.8632	0.8368	0.8421
KNN(5)	0.8526	0.8263	0.8263
KNN(6)	0.8316	0.8368	0.8316
KNN(7)	0.8316	0.8368	0.8421
KNN(8)	0.8474	0.8474	0.8421
KNN(9)	0.8316	0.8474	0.8368
KNN(10)	0.8211	0.8526	0.8474
KNN(11)	0.8368	0.8316	0.8368
KNN(12)	0.8105	0.8316	0.8316
MLP(100)[10]	0.9158	0.9211	0.9105
RandomForest[10]	**0.9316**	0.8895	0.9105
SVC[10]	0.9053	0.8947	0.8947
GaussianNB[10]	0.8579	0.8737	0.8632

Table 4.9.: *LOOCV on embeddings extracted from the model $\mathcal{M}_{\text{entropic-openset}}$. Embedding A trained on RandomForest[10] achieved the highest accuracy of* 93.16%.

Algorithm	Embedding A	Embedding B	Embedding C
KNN(1)	0.8316	0.8158	0.8158
KNN(2)	0.8474	0.8263	0.8316
KNN(3)	0.8632	0.8263	0.8263
KNN(4)	0.8684	0.8579	0.8632
KNN(5)	0.8474	0.8526	0.8526
KNN(6)	0.8316	0.8316	0.8368
KNN(7)	0.8158	0.8316	0.8316
KNN(8)	0.8158	0.8158	0.8211
KNN(9)	0.8211	0.8211	0.8211
KNN(10)	0.8211	0.8000	0.8000
KNN(11)	0.8105	0.7789	0.7789
KNN(12)	0.7842	0.7895	0.7895
MLP(100)[10]	**0.9263**	0.8947	0.9053
RandomForest[10]	0.9000	0.8842	0.8895
SVC[10]	0.8632	0.8737	0.8737
GaussianNB[10]	0.7000	0.8526	0.7158

Table 4.10.: *LOOCV on embeddings extracted from the model $\mathcal{M}_{\text{background-as-class}}$. Embedding A trained on MLP(100)[10] achieved the highest accuracy of 92.63%.*

4.4. Analysing the results

We have shown that segmenting unknown gestures can be effectively performed with the Entropic Open-Set loss using pronounced thresholding and Cross-Entropy loss by introducing a background class and thresholding on it. According to our overlapping error criteria, we get an accuracy rate of 95% on the segmentation. Alternatively, if we look at missing gestures and missing gaps we see that we have 1 missing gesture and 4 missing gaps, meaning we have $\frac{51}{52} = 98\%$ recognition of gestures and $\frac{47}{51} = 92\%$ recognition of gaps.

Further, the best performing embeddings were all extracted from $\mathcal{M}_{\text{standard}}$, indicating that embeddings are best extracted from a regularly trained MS-G3D model using the Cross-Entropy loss. We suspect that this is the case because the other models do not only create good classification embeddings but are also burdened by distinguishing meaningful gestures from meaningless in-between movements.

In the 1-shot scenario, Embedding A outperformed the other embeddings with nearest neighbour search, independent of which functions were used to train the embeddings. Similarly, in the 4-shot scenario, Embedding A, trained on ten aggregated neural networks with hidden layer size 100, outperformed the other embeddings, independent of which loss functions were used to train the embeddings. For this reason, we conclude that for a small number of samples, values extracted from the latent space one layer before the Softmax layer, ie. Embedding A, form the best embeddings. It is harder to discern which machine learning method and embedding achieved the best performance when evaluating the entire ninja dataset.

Ultimately, the pipeline can be used as-is in cases where classification performance of 87% per gesture for 4 samples per class is acceptable but needs further improvement for more demanding applications.

5 Conclusion and Future Work

5.1. Conclusion

In conclusion, we proposed a pipeline for future gesture recognition work with unknown gestures at runtime, as seen in Figure 3.1. We have shown that segmenting unknown gestures can be effectively performed with the Entropic Open-Set loss using pronounced thresholding and Cross-Entropy loss by introducing a background class and thresholding on it. According to our overlapping error criteria, we get an accuracy rate of 95% on the segmentation. Alternatively, if we look at missing gestures and missing gaps we see that we have 1 missing gesture and 4 missing gaps, meaning we have $\frac{51}{52} \approx 98\%$ recognition of gestures and $\frac{47}{51} \approx 92\%$ recognition of gaps.

Further, we have proposed a method to extract effective embeddings A, B and C from MS-G3D models $\mathcal{M}_{\text{standard}}$, $\mathcal{M}_{\text{entropic-openset}}$ and $\mathcal{M}_{\text{background-as-class}}$. This extraction should also be extendable to other graph convolutional neural networks.

The best performing embeddings were all extracted from $\mathcal{M}_{\text{standard}}$, indicating that embeddings are best extracted from a regularly trained MS-G3D model using the Cross-Entropy loss. We suspect that this is the case because the other models do not only create good classification embeddings but are also burdened by distinguishing meaningful gestures from meaningless in-between movements.

In the 1-shot scenario, Embedding A outperformed the other embeddings with nearest neighbour search, independent of which functions were used to train the embeddings. Similarly, in the 4-shot scenario, Embedding A, trained on ten aggregated neural networks with hidden layer size 100, outperformed the other embeddings, independent of which loss functions were used to train the embeddings. For this reason, we conclude that for a small number of samples, values extracted from the latent space one layer before the Softmax layer, ie. Embedding A, form the best embeddings. It is harder to discern which machine learning method and embedding achieved the best performance when evaluating the entire ninja dataset. However, the best performing embeddings were all extracted from $\mathcal{M}_{\text{standard}}$.

Finally, the pipeline can be used as-is in cases where classification performance of 87% per gesture for 4 samples per class is acceptable but needs further improvement for more demanding applications, which brings us to the Future Work section.

5.2. Future Work

Any improvement in the pipeline 3.1 will lead to better unknown gesture classification accuracies. Concretely, we suggest improving models such as MediaPipe, so hand skeletons are present in all frames. Changing from OpenPose to MediaPipe leads to a performance increase of 66.13% to 87.08% and will likely contribute the most to the model's performance. Further, a temporal convolution over the output probability values of the respected classes might significantly increase segmentation performance instead of fitting τ and κ variables. However, to train this without overfitting, a larger dataset than the continuous ninja dataset is required. Lastly, we suggest using an autoencoder to generate better embeddings, where the encoder has a structure similar to MS-G3D or another graph convolutional neural network.

Our suggested segmentation techniques can be tested on other neural networks that use a Cross-Entropy loss. Concretely, future work can apply this method to other neural networks used in gesture recognition or even to research fields like video segmentation. In addition, future work can extract embeddings from the skeletal reconstruction networks directly, so the embedding step can already be applied when reconstructing the skeleton and be skipped later in the pipeline.

Lastly, alternatives to Softmax [Banerjee et al. 2020] could be combined with the Entropic Open-Set loss method to get better performance.

A

Information For The Few (Appendix)

```
err = 0
totposserr = 0
predstate = 0
truthstate = 0
for row in range(len(predictions)):

   pred = predictions[row]
   truth = 0
   if row+shiftoffset<len(ground_truth):
      truth = ground_truth[row+shiftoffset]

   if pred == 0:
      if predstate == 1:
         err += 1
      elif predstate == 2:
         totposserr += 1
      if predstate != -2:
         predstate = -1

   if truth == 0:
      if truthstate == 1:
         err += 1
      elif truthstate == 2:
         totposserr += 1
      if truthstate != -2:
         truthstate = -1

   if pred == 1:
      if predstate == -1:
         err += 1
      elif predstate == -2:
         totposserr += 1
      if predstate != 2:
         predstate = 1

   if truth == 1:
      if truthstate == -1:
         err += 1
      elif truthstate == -2:
         totposserr += 1
      if truthstate != 2:
         truthstate = 1

   if pred == 1 and truth == 1 and predstate == 1 and truthstate == 1:
      predstate = 2
      truthstate = 2

   if pred == 0 and truth == 0 and predstate == -1 and truthstate == -1:
      predstate = -2
      truthstate = -2


relativeerror = err/totposserr
```

Figure A.1.: *Overlapping error criteria based on which τ, the threshold, and κ, the minimum dimensionality for a gesture to be meaningful, were fitted with grid-search. The error counts the number of correct prediction/ground truth gaps/gesture overlaps by starting at frame 0 and matching off gestures and gaps from the predicted values with the ground truth values. If any gesture or gap did not find a match it is counted as an error.*

```
class EntropicOpenSetLoss(torch.nn.CrossEntropyLoss):
   __constants__ = ['ignore_index', 'reduction']
   ignore_index: int

   def __init__(self, weight: Optional[Tensor] = None, size_average=None,
       ignore_index: int = -100,
      reduce=None, reduction: str = 'mean') -> None:

      self.ignore_index = ignore_index
      super(nn.CrossEntropyLoss, self).__init__(weight, size_average,
         reduce, reduction)

   def forward(self, input: Tensor, target: Tensor) -> Tensor:
      full_target = [i for i in range(0,backgroundid)]
      full_target_t = torch.tensor(full_target, requires_grad=False, dtype=
         torch.long).cuda()
      batch_size = input.shape[0]
      loss = 0
      for tidx, tid in enumerate(target):
         if tid == backgroundid: # background case
            loss += 1.0/batch_size
                 * 1.0/backgroundid
                 * nn.functional.cross_entropy(
                       input[tidx].repeat(backgroundid,1),
                       full_target_t,
                       weight=self.weight,
                       ignore_index=self.ignore_index,
                       reduction=self.reduction)
         else: # normal case
            loss += 1.0/batch_size
                  * nn.functional.cross_entropy(
                        input[tidx].unsqueeze(0),
                        target[tidx].unsqueeze(0),
                        weight=self.weight,
                        ignore_index=self.ignore_index,
                        reduction=self.reduction)

      return loss
```

Figure A.2.: *An optimised implementation for the Entropic OpenSet loss. Note that the indices $\{0,\ldots,backgroundid-1\}$ represent classes of meaningful gestures while $\{backgroundid\}$ denotes the background class.*

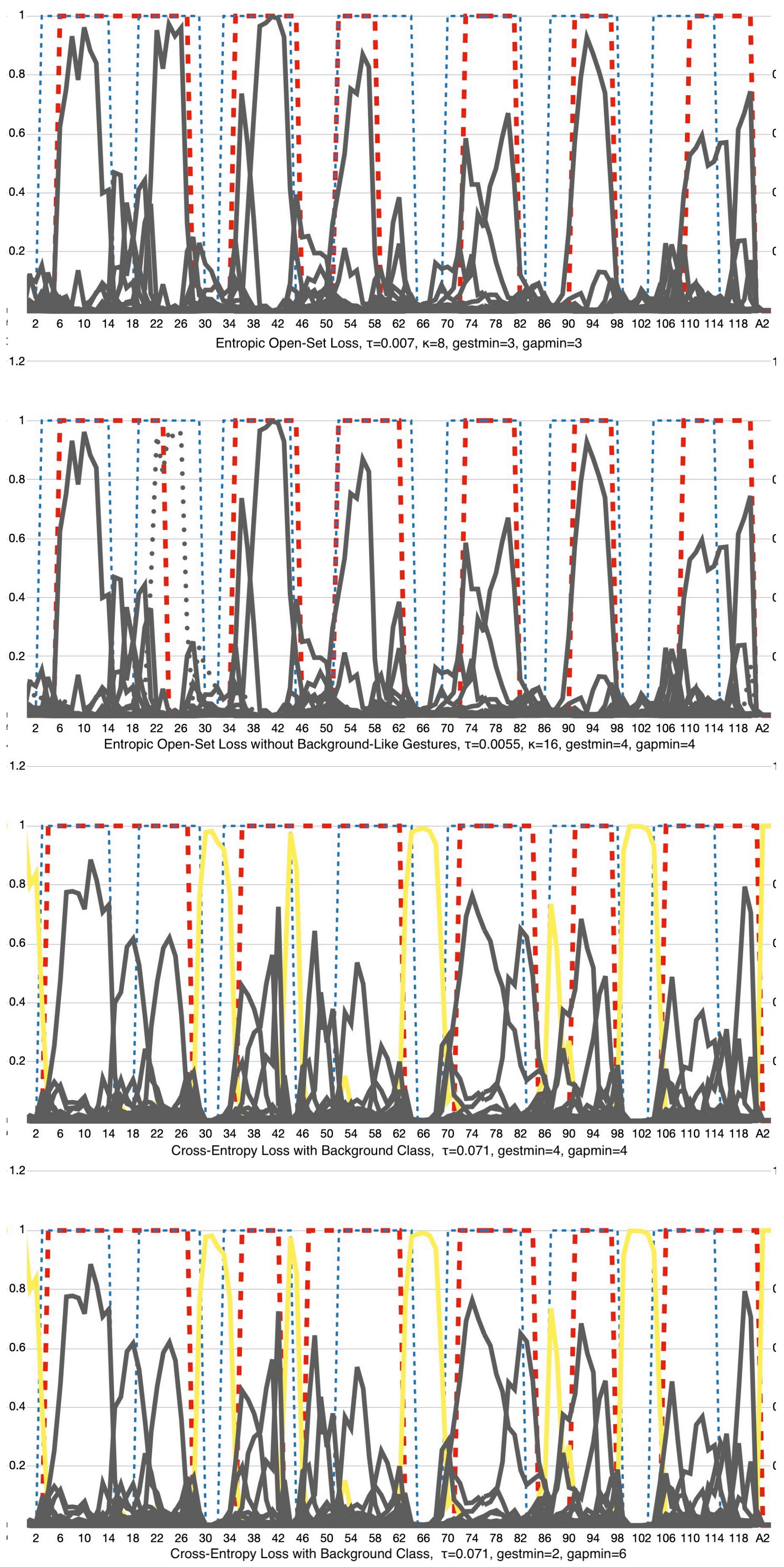


Figure A.3.: *Continuous ninja sentence A1 with segmentation predictions in dashed red and ground-truth segmentations in dashed blue. The background class is denoted in solid yellow where available and the other output probability values of each model are denoted in grey.*

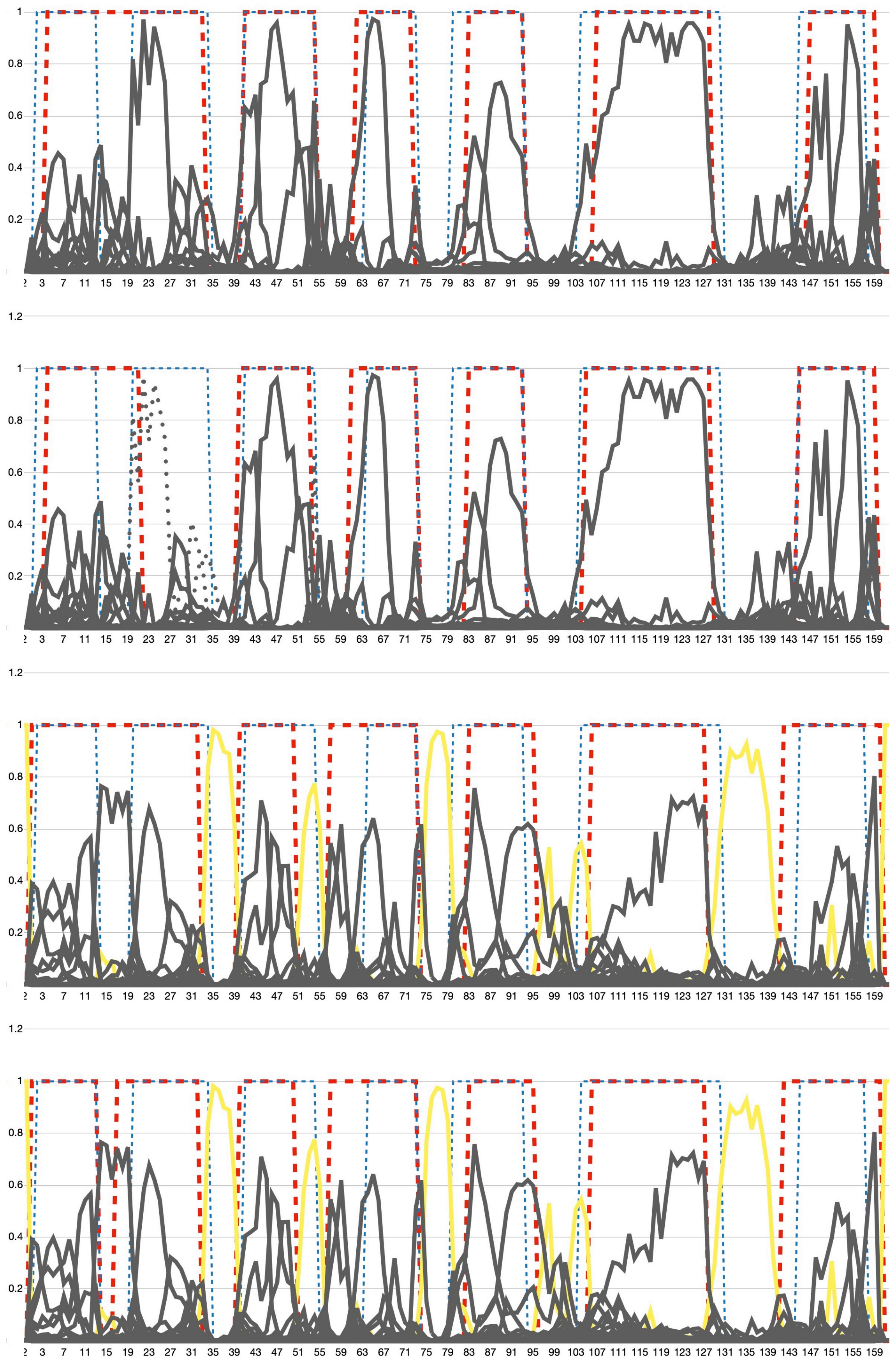


Figure A.4.: *Continuous ninja sentence A2 with segmentation predictions in dashed red and ground-truth segmentations in dashed blue. The background class is denoted in solid yellow where available and the other output probability values of each model are denoted in grey.*

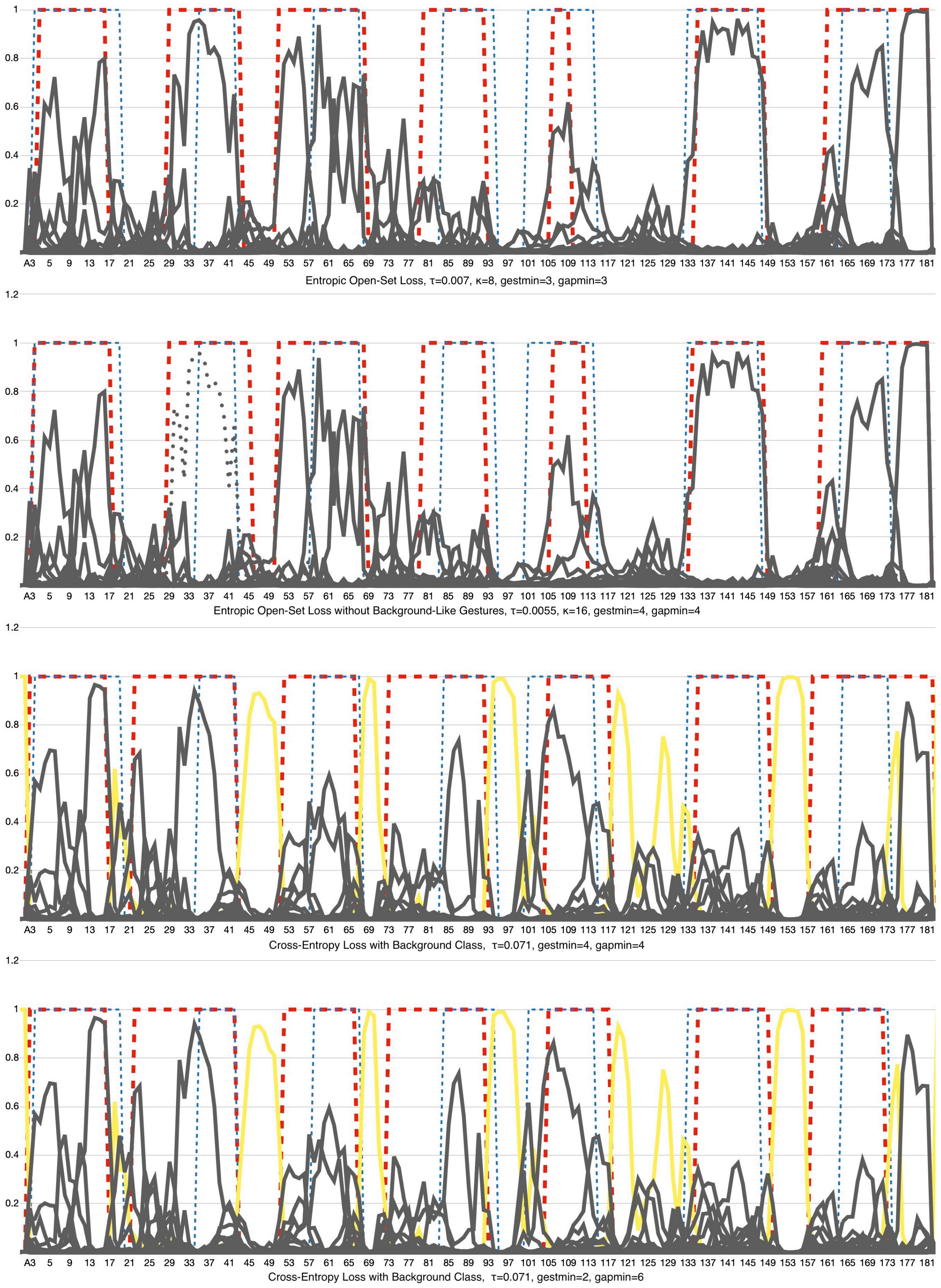


Figure A.5.: *Continuous ninja sentence A3 with segmentation predictions in dashed red and ground-truth segmentations in dashed blue. The background class is denoted in solid yellow where available and the other output probability values of each model are denoted in grey.*

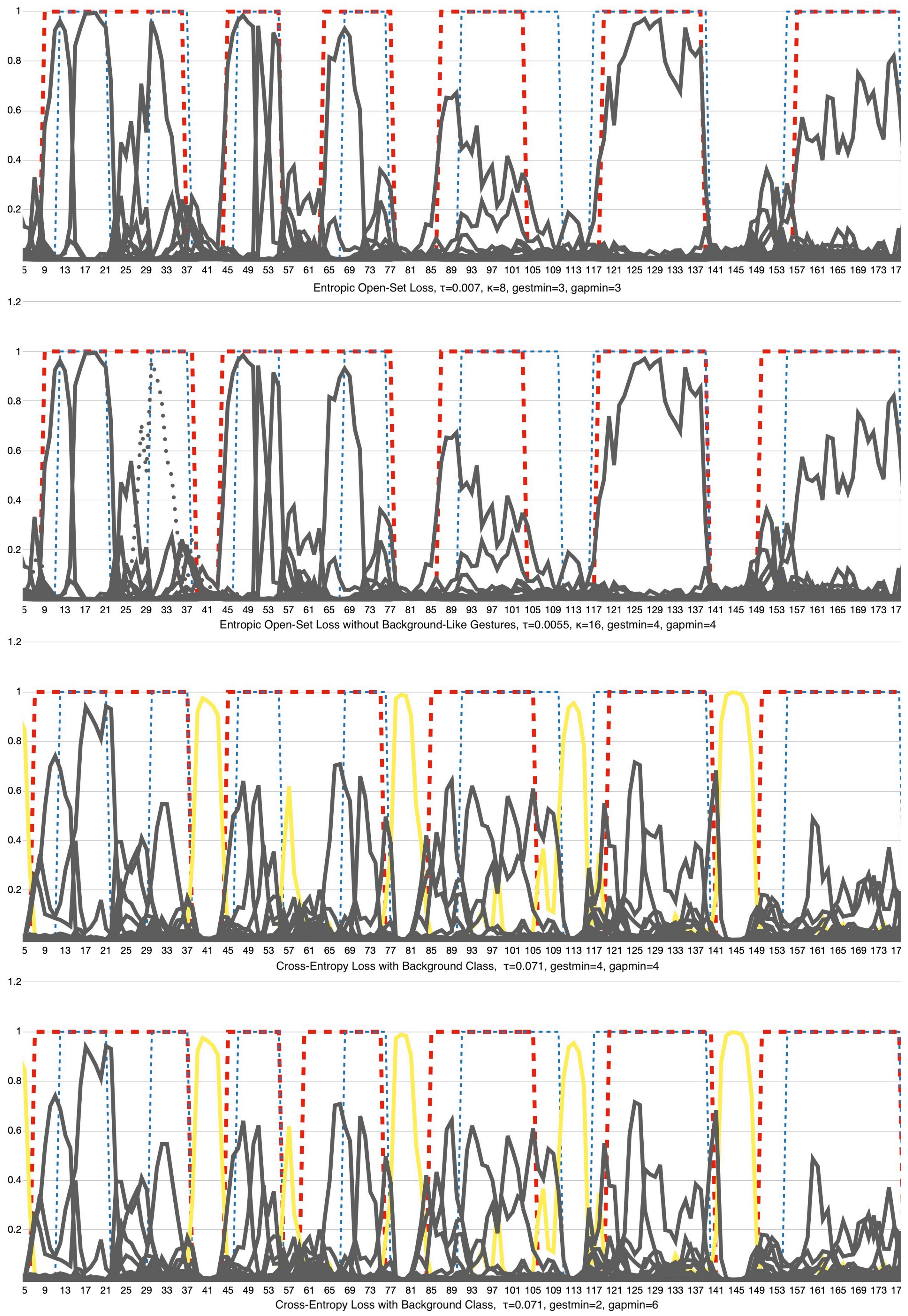


Figure A.6.: *Continuous ninja sentence A4 with segmentation predictions in dashed red and ground-truth segmentations in dashed blue. The background class is denoted in solid yellow where available and the other output probability values of each model are denoted in grey.*

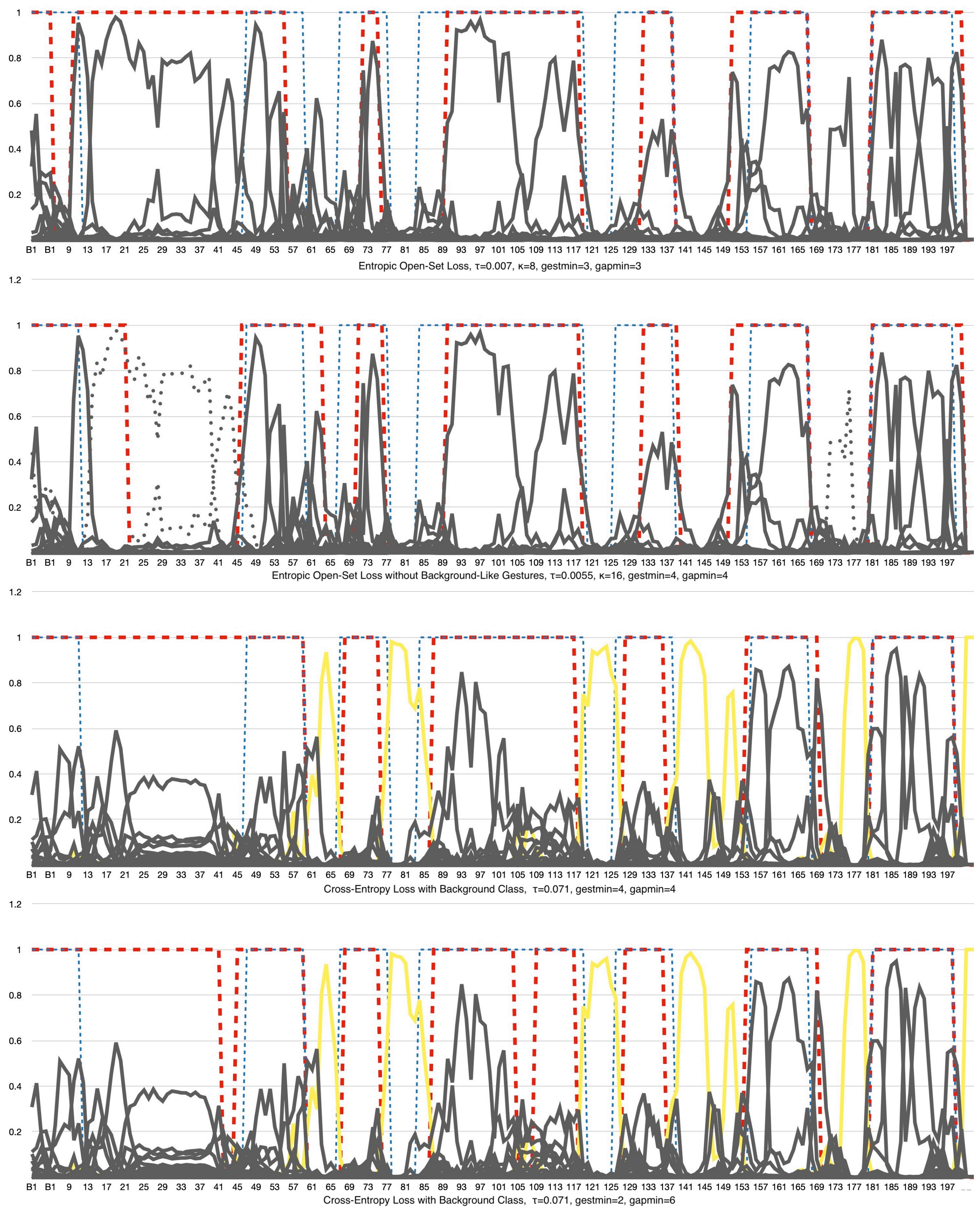


Figure A.7.: *Continuous ninja sentence B1 with segmentation predictions in dashed red and ground-truth segmentations in dashed blue. The background class is denoted in solid yellow where available and the other output probability values of each model are denoted in grey.*

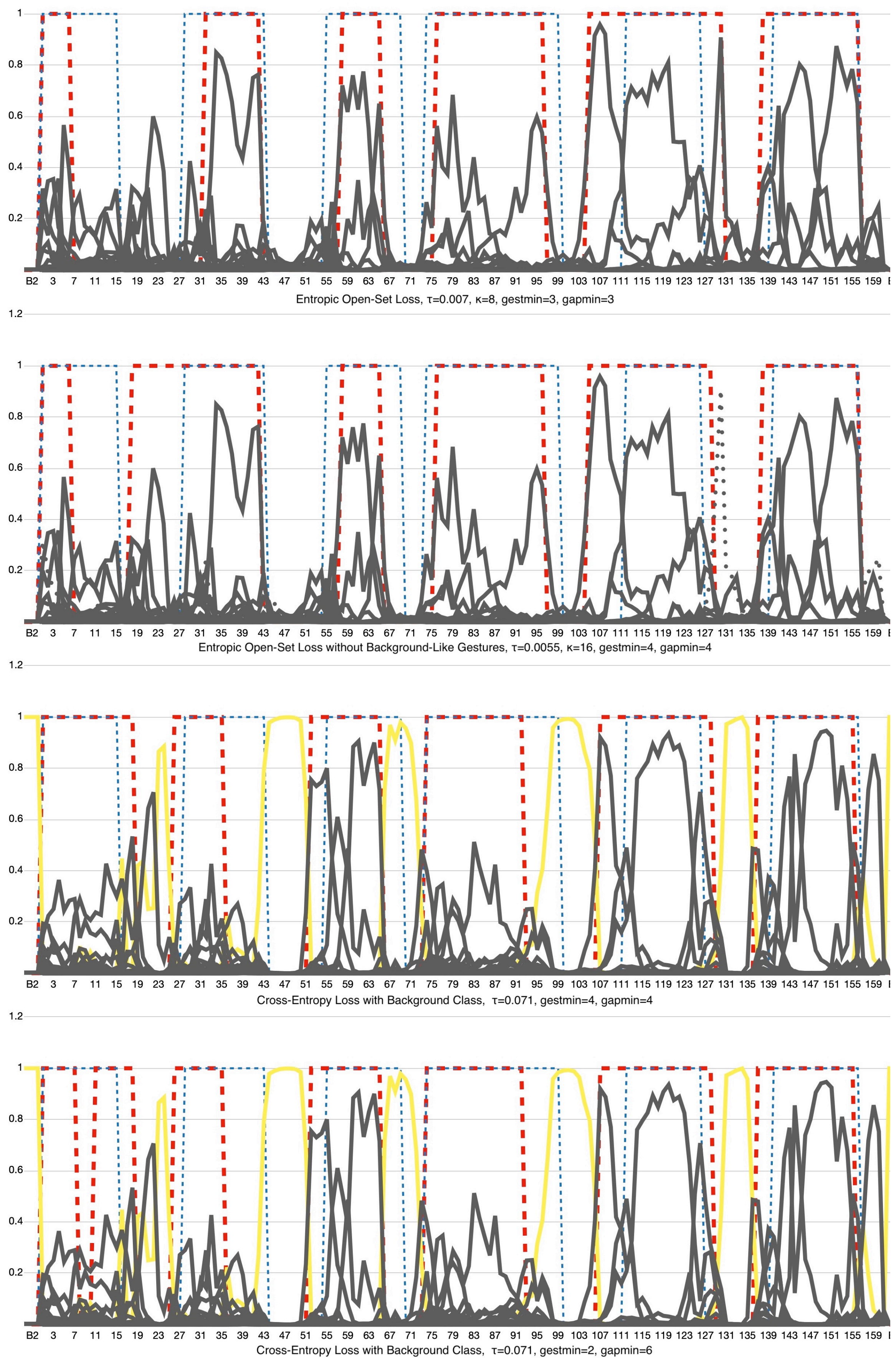


Figure A.8.: *Continuous ninja sentence B2 with segmentation predictions in dashed red and ground-truth segmentations in dashed blue. The background class is denoted in solid yellow where available and the other output probability values of each model are denoted in grey.*

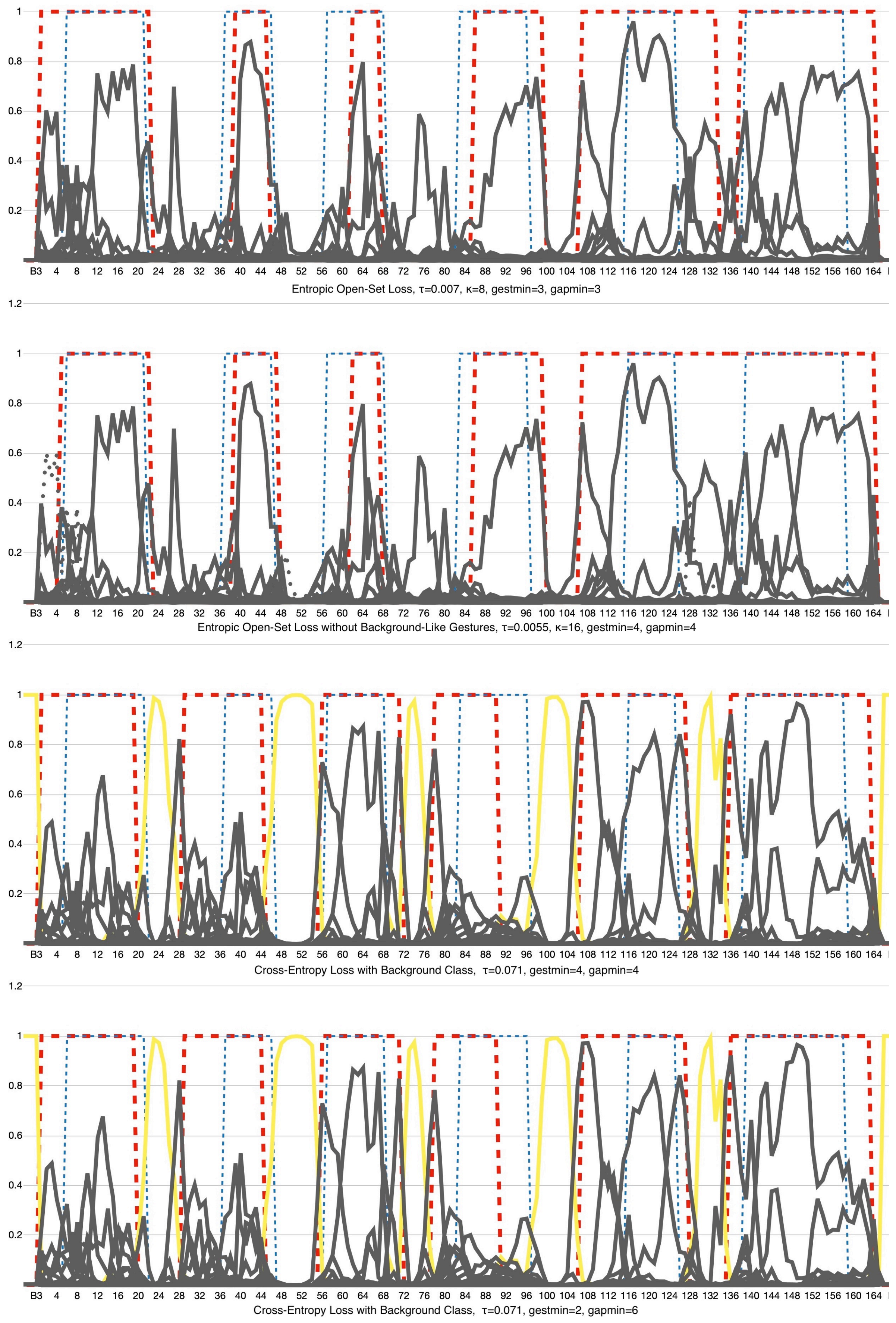


Figure A.9.: *Continuous ninja sentence B3 with segmentation predictions in dashed red and ground-truth segmentations in dashed blue. The background class is denoted in solid yellow where available and the other output probability values of each model are denoted in grey.*

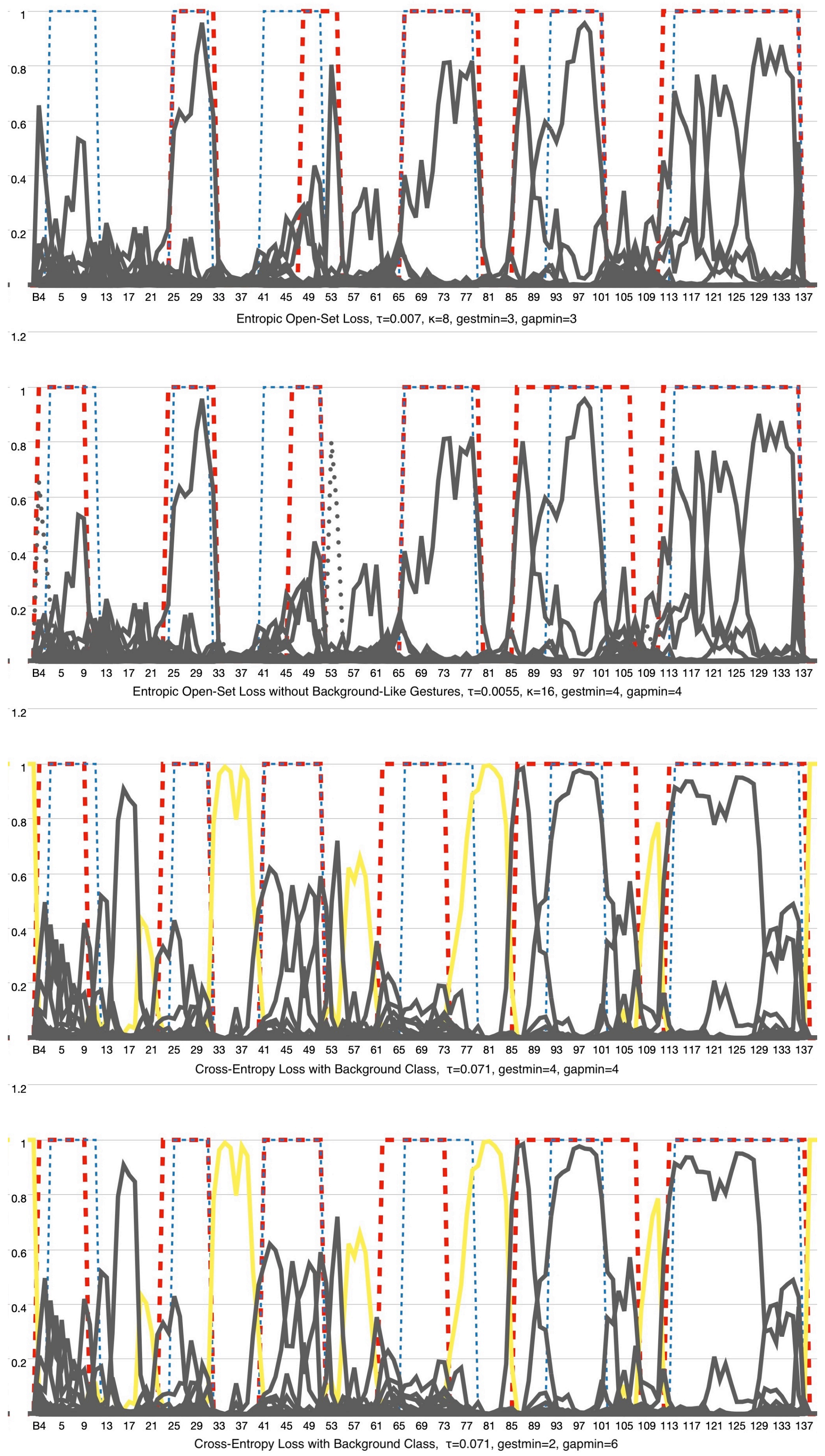


Figure A.10.: *Continuous ninja sentence B4 with segmentation predictions in dashed red and ground-truth segmentations in dashed blue. The background class is denoted in solid yellow where available and the other output probability values of each model are denoted in grey.*

Bibliography

ADALOGLOU, N., CHATZIS, T., PAPASTRATIS, I., STERGIOULAS, A., PAPADOPOULOS, G. T., ZACHAROPOULOU, V., XYDOPOULOS, G. J., ATZAKAS, K., PAPAZACHARIOU, D., AND DARAS, P. A Comprehensive Study on Sign Language Recognition Methods. t. *arXiv preprint arXiv:2007.12530*.

BANERJEE, K., C, V. P., GUPTA, R. R., VYAS, K., H, A., AND MISHRA, B., Exploring Alternatives to Softmax Function. t.

BRONSTEIN, M. M., BRUNA, J., COHEN, T., AND VELIČKOVIĆ, P., Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. t.

CAO, Z., HIDALGO, G., SIMON, T., WEI, S., AND SHEIKH, Y. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. t. *CoRR abs/1812.08008*.

CAO, C., LAN, C., ZHANG, Y., ZENG, W., LU, H., AND ZHANG, Y. Skeleton-Based Action Recognition with Gated Convolutional Neural Networks. t. *IEEE Trans. on Cir. and Sys. for Video Technology (to appear)* (February).

CHEN, Y., ZHANG, Z., YUAN, C., LI, B., DENG, Y., AND HU, W., Channel-wise Topology Refinement Graph Convolution for Skeleton-Based Action Recognition. t.

CHENG, H., YANG, L., AND LIU, Z. Survey on 3D hand gesture recognition. t. *IEEE transactions on circuits and systems for video technology 26*, 9, 1659–1673.

CHENG, K., ZHANG, Y., CAO, C., SHI, L., CHENG, J., AND LU, H. Decoupling gcn with dropgraph module for skeleton-based action recognition. t. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, Springer, 536–553.

CHENG, K., ZHANG, Y., HE, X., CHEN, W., CHENG, J., AND LU, H. Skeleton-Based Action Recognition With Shift Graph Convolutional Network. t. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 180–189.

DARDAS, N. H., AND GEORGANAS, N. D. Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. t. *IEEE Transactions on Instrumentation and measurement 60*, 11, 3592–3607.

DHAMIJA, A. R., GÜNTHER, M., AND BOULT, T. E., Reducing Network Agnostophobia. t.

VIVA (Vision for Intelligent Vehicles and Applications). t.

DU, Y., WANG, W., AND WANG, L. Hierarchical recurrent neural network for skeleton based action recognition. t. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1110–1118.

DUAN, H., ZHAO, Y., CHEN, K., SHAO, D., LIN, D., AND DAI, B. Revisiting Skeleton-based Action Recognition (PoseC3D). t. *CoRR abs/2104.13586*.

FEICHTENHOFER, C., FAN, H., MALIK, J., AND HE, K., SlowFast Networks for Video Recognition. t.

FELS, S., AND HINTON, G. Glove-Talk: a neural network interface between a data-glove and a speech synthesizer. t. *IEEE Transactions on Neural Networks 4*, 1, 2–8.

GENG, C., HUANG, S.-J., AND CHEN, S. Recent Advances in Open Set Recognition: A Survey. t. *IEEE Transactions on Pattern Analysis and Machine Intelligence 43*, 10 (Oct), 3614–3631.

GHOTKAR, A. S., KHATAL, R., KHUPASE, S., ASATI, S., AND HADAP, M. Hand gesture recognition for indian sign language. t. In *2012 International Conference on Computer Communication and Informatics*, IEEE, 1–4.

GRIMES, G. J., Digital data entry glove interface device. t. US Patent Nr. US4414537A.

HE, K., ZHANG, X., REN, S., AND SUN, J., Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. t.

HIPTMAIR, P. R., ARBENZ, P. P., AND GRADINARU, D. V., Numerical Methods for Computational Science and Engineering, ETH Lecture 401-0663-00L. t.

JEONG, M. H., KUNO, Y., SHIMADA, N., AND SHIRAI, Y. Two-hand gesture recognition using coupled switching linear model. t. In *Object recognition supported by user interaction for service robots*, vol. 3, IEEE, 529–532.

KE, Q., BENNAMOUN, M., AN, S., SOHEL, F. A., AND BOUSSAÏD, F. A New Representation of Skeleton Sequences for 3D Action Recognition. t. *CoRR abs/1703.03492*.

KHATIB, O. A unified approach for motion and force control of robot manipulators: The operational space formulation. t. *IEEE Journal on Robotics and Automation 3*, 1, 43–53.

KIM, T. S., AND REITER, A. Interpretable 3D Human Action Analysis with Temporal Convolutional Networks. t. *CoRR abs/1704.04516*.

KIPF, T. N., AND WELLING, M., Semi-Supervised Classification with Graph Convolutional Networks. t.

KISHIMOTO, M. 1999. *Naruto*.

KISHORE, P., AND KUMAR, P. R. A model for real time sign language recognition system. t. *International Journal of Advanced Research in Computer Science and Software Engineering 2*, 6.

KONSTANTINIDIS, D., DIMITROPOULOS, K., AND DARAS, P. SIGN LANGUAGE RECOGNITION BASED ON HAND AND BODY SKELETAL DATA. t. In *2018 - 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, 1–4.

KRAMER, J. P., LINDENER, P., AND GEORGE, W. R., Communication system for deaf, deaf-blind, or non-vocal individuals using instrumented glove. t, Sept. 10. US Patent 5,047,952.

LEE, I., KIM, D., KANG, S., AND LEE, S. Ensemble Deep Learning for Skeleton-Based Action Recognition Using Temporal Sliding LSTM Networks. t. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 1012–1020.

LEI, Q., ZHANG, H., XIA, Z., YANG, Y., HE, Y., AND LIU, S. Applications of hand gestures recognition in industrial robots: a review . t. In *Eleventh International Conference on Machine Vision (ICMV 2018)*, SPIE, A. Verikas, D. P. Nikolaev, P. Radeva, and J. Zhou, Eds., vol. 11041, International Society for Optics and Photonics, 455 – 465.

LI, C., ZHONG, Q., XIE, D., AND PU, S. Co-occurrence Feature Learning from Skeleton Data for Action Recognition and Detection with Hierarchical Aggregation. t. *CoRR abs/1804.06055*.

LI, M., CHEN, S., CHEN, X., ZHANG, Y., WANG, Y., AND TIAN, Q., Actional-Structural Graph Convolutional Networks for Skeleton-based Action Recognition. t.

LI, S., LI, W., COOK, C., AND GAO, Y., Deep Independently Recurrent Neural Network (IndRNN). t.

LIU, J., SHAHROUDY, A., XU, D., AND WANG, G. Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition. t. *CoRR abs/1607.07043*.

LIU, M., LIU, H., AND CHEN, C. Enhanced skeleton visualization for view invariant human action recognition. t. *Pattern Recognition 68*, 346–362.

LIU, J., SHAHROUDY, A., PEREZ, M., WANG, G., DUAN, L.-Y., AND KOT, A. C. NTU RGB+D 120: A large-scale benchmark for 3D human activity understanding. t. *IEEE Transactions on Pattern Analysis and Machine Intelligence 42*, 10, 2684–2701.

LIU, Z., ZHANG, H., CHEN, Z., WANG, Z., AND OUYANG, W. Disentangling and Unifying Graph Convolutions for Skeleton-Based Action Recognition. t. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 143–152.

LUGARESI, C., TANG, J., NASH, H., MCCLANAHAN, C., UBOWEJA, E., HAYS, M., ZHANG, F., CHANG, C.-L., YONG, M. G., LEE, J., CHANG, W.-T., HUA, W., GEORG, M., AND GRUNDMANN, M., MediaPipe: A Framework for Building Perception Pipelines. t.

MARO, J.-M., IENG, S.-H., AND BENOSMAN, R. Event-Based Gesture Recognition With Dynamic Background Suppression Using Smartphone Computational Capabilities. t. *Frontiers in Neuroscience 14*, 275.

MITRA, S., AND ACHARYA, T. Gesture Recognition: A Survey. t. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 37*, 3, 311–324.

MOLCHANOV, P., YANG, X., GUPTA, S., KIM, K., TYREE, S., AND KAUTZ, J. Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks. t. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4207–4215.

NIAIS, ., Awesome Skeleton-based Action Recognition Github page. t.

NINTENDO, Nintendogs manual. Nintendo DS. t.

PAUL, R. 1981. *Robot Manipulators: Mathematics, Programming, and Control : the Computer Control of Robot Manipulators*. Artificial Intelligence Series. MIT Press.

PENG, W., HONG, X., CHEN, H., AND ZHAO, G., Learning Graph Convolutional Network for Skeleton-based Human Action Recognition by Neural Searching. t.

RAUTARAY, S. S. Real time hand gesture recognition system for dynamic applications. t. *International Journal of UbiComp (IJU) 3*, 1.

RICHARD, W. A survey of Gesture Recognition Techniques Technical Report. t. *Trinity College, Dublin*, 6.

SARKAR, A. R., SANYAL, G., AND MAJUMDER, S. Hand gesture recognition systems: a survey. t. *International Journal of Computer Applications 71*, 15.

SCHNYDER, J. J. A., AND GUENTHER, M., Deep Adversarial Training for Teaching Networks to Reject Unknown Inputs. t, July.

SHAHROUDY, A., LIU, J., NG, T.-T., AND WANG, G. NTU RGB+D: A large scale dataset for 3D human activity analysis. t. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1010–1019.

SHAHROUDY, A., LIU, J., NG, T.-T., AND WANG, G. NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis. t. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1010–1019.

SHI, L., ZHANG, Y., CHENG, J., AND LU, H. Skeleton-Based Action Recognition With Directed Graph Neural Networks. t. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7904–7913.

SHI, L., ZHANG, Y., CHENG, J., AND LU, H., Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition. t.

SI, C., JING, Y., WANG, W., WANG, L., AND TAN, T., Skeleton-Based Action Recognition with Spatial Reasoning and Temporal Stack Learning. t.

SI, C., CHEN, W., WANG, W., WANG, L., AND TAN, T., An Attention Enhanced Graph Convolutional LSTM Network for Skeleton-Based Action Recognition. t.

SONG, S., LAN, C., XING, J., ZENG, W., AND LIU, J., An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data. t.

SONG, Y.-F., ZHANG, Z., AND WANG, L. Richly Activated Graph Convolutional Network for Action Recognition with Incomplete Skeletons. t. In *2019 IEEE International Conference on Image Processing (ICIP)*, 1–5.

SONG, Y.-F., ZHANG, Z., SHAN, C., AND WANG, L. Stronger, Faster and More Explainable: A Graph Convolutional Baseline for Skeleton-based Action Recognition. t. *Proceedings of the 28th ACM International Conference on Multimedia* (Oct).

SONG, Y., ZHANG, Z., SHAN, C., AND WANG, L. Constructing Stronger and Faster Baselines for Skeleton-based Action Recognition. t. *CoRR abs/2106.15125*.

STURMAN, D., AND ZELTZER, D. A survey of glove-based input. t. *IEEE Computer Graphics and Applications 14*, 1, 30–39.

SUN, K., XIAO, B., LIU, D., AND WANG, J. Deep High-Resolution Representation Learning for Human Pose Estimation (HRNet). t. *CoRR abs/1902.09212*.

TANG, Y.-M., AND HUI, K.-C. Simulating tendon motion with axial mass-spring system. t. *Computers & Graphics 33*, 2, 162–172.

TANG, Y., TIAN, Y., LU, J., LI, P., AND ZHOU, J. Deep Progressive Reinforcement Learning for Skeleton-Based Action Recognition. t. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5323–5332.

THAKKAR, K., Part-based Graph Convolutional Network for Action Recognition. t.

TOLENTINO, R. Application of Template Matching Algorithm for Dynamic Gesture Recognition of American Sign Language Finger Spelling and Hand Gesture. t. *Asia Pacific Journal of Multidisciplinary Research 2* (08), 154.

TOMEI, P. A simple PD controller for robots with elastic joints. t. *IEEE Transactions on automatic control 36*, 10, 1208–1213.

TRINDADE, P., LOBO, J., AND BARRETO, J. P. Hand gesture recognition using color and depth images enhanced with hand angular pose data. t. In *2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, IEEE, 71–76.

VEMULAPALLI, R., ARRATE, F., AND CHELLAPPA, R. Human Action Recognition by Representing 3D Skeletons as Points in a Lie Group. t. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE Computer Society, Los Alamitos, CA, USA, 588–595.

WANG, H., AND WANG, L. Modeling Temporal Dynamics and Spatial Configurations of Actions Using Two-Stream Recurrent Neural Networks. t. *CoRR abs/1704.02581.*

WANG, H., AND WANG, L. Beyond Joints: Learning Representations From Primitive Geometries for Skeleton-Based Action Recognition and Detection. t. *IEEE Transactions on Image Processing 27*, 9, 4382–4394.

WANG, M., NI, B., AND YANG, X. Learning Multi-View Interactional Skeleton Graph for Action Recognition. t. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1.

WEN, Y.-H., GAO, L., FU, H., ZHANG, F.-L., AND XIA, S. Graph CNNs with Motif and Variable Temporal Block for Skeleton-Based Action Recognition. t. *Proceedings of the AAAI Conference on Artificial Intelligence 33*, 01 (Jul.), 8989–8996.

WILSON, A., AND BOBICK, A. Parametric hidden Markov models for gesture recognition. t. *IEEE Transactions on Pattern Analysis and Machine Intelligence 21*, 9, 884–900.

WU, C., WU, X.-J., AND KITTLER, J. Spatial Residual Layer and Dense Connection Block Enhanced Spatial Temporal Graph Convolutional Network for Skeleton-Based Action Recognition. t. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 1740–1748.

XIE, C., LI, C., ZHANG, B., CHEN, C., HAN, J., ZOU, C., AND LIU, J., Memory Attention Networks for Skeleton-based Action Recognition. t.

YAN, S., XIONG, Y., AND LIN, D. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. t. *CoRR abs/1801.07455.*

YANG, J., AND XU, Y. Hidden markov model for gesture recognition. t. Tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST.

ZAHEER, M., KOTTUR, S., RAVANBAKHSH, S., POCZOS, B., SALAKHUTDINOV, R., AND SMOLA, A., Deep Sets. t.

ZHANG, P., LAN, C., XING, J., ZENG, W., XUE, J., AND ZHENG, N. View Adaptive Recurrent Neural Networks for High Performance Human Action Recognition from Skeleton Data. t. *CoRR abs/1703.08274*.

ZHANG, P., LAN, C., XING, J., ZENG, W., XUE, J., AND ZHENG, N., View Adaptive Neural Networks for High Performance Skeleton-based Human Action Recognition. t.

ZHANG, P., LAN, C., ZENG, W., XING, J., XUE, J., AND ZHENG, N., Semantics-Guided Neural Networks for Efficient Skeleton-Based Human Action Recognition. t.

ZHOU, X. S., AND HUANG, T. Small sample learning during multimedia retrieval using BiasMap. t. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, I–I.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Gesture2Vec - Detecting and recognising unknown gestures for classification purposes

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
De Keyser	Kevin Joël Philippe

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 24th of September 2021	[illegible]

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.