Microgesture Hand Classification from Egocentric Viewpoints

Jeremy J. Hartmann
University of Waterloo

Hemant Bhaskar Surale
University of Waterloo

***Abstract*—The research focus of hand gesture recognition has moved from detecting coarse gestures to tracking individual finger positions – often termed as subtle or micro hand gestures. Robust recognition of such hand microgestures opens up huge potentials for designers to create precise, appealing, and expressive interaction techniques. Yet, past research has largely involved in classifying coarse gestures with 4-channel RGB-D image data from depth cameras, which limits the applicability to mobile form factors often equipped with only a monocular RGB camera. Our paper presents a micro hand gesture classifier which recognizes four different subtle gestures with the thumb placed on different areas of the index finger. We trained different types of neural networks on synthetic hand depth images and tested with input from a RGB camera with an infrared filter. Our best classifier could achieved overall accuracy up to 57%. Finally, we present a comparative analysis of these different neural architectures for the gesture recognition task and contribute a micro hand gesture dataset. We also explore architecture design guidelines and future extensions to our work.**

***Keywords*—CS870, Neural Networks, classification, CNN, microgestures.**

I. Introduction

In augmented or virtual reality settings, user interaction with virtual objects are commonly enabled by instrumenting physical controllers [1], [2] or wearable gloves [3], [4], [5]. These devices may be advantageous in providing accurate tracking input, but they can be cumbersome to carry and not as expressive and dexterous as the human hand. As a result, tracking bare hands with non-intrusive sensors has become an increasingly active research area [6], [7]. Furthermore, past research has primarily focused on coarse hand gestures [8], [9], [10], partly because of the imprecise and non-robust sensing technologies. Devices like Microsoft HoloLens, Leap Motion, Meta 2, and PrimeSense can accurately track coarse hand gestures, but suffer from finger occlusion issues and also, lacks robust fingertip tracking.

However, for rigorous tasks such as keyboard input or interactions with user interface elements, coarse hand gestures are not suitable. Instead, they require a fine grain tracking of fingers, or even fingertips, to facilitate accurate control. Nonetheless, with the advent of better learning algorithms (Deep learning) and high resolution cameras, we hypothesize that more precise hand gestures, such as single handed microgestures [11], can be recognized using an infrared camera or a depth image or a RGB image. Specifically, in this paper, we focus on building a classifier for microgestures that would work with grayscale images, similar to the images produced with a smartphone camera. Microgesture recognition with smartphone camera would enrich interaction vocabulary as well as empower application developers to create novel interactive experiences.

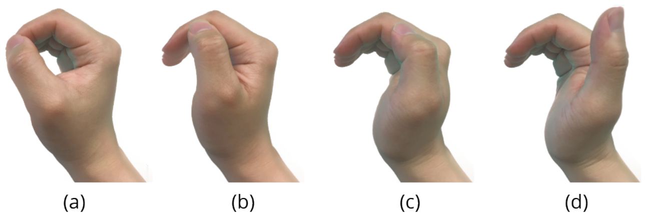

Fig. 1: Set of microgestures where the point of contact between thumb and index finger is the minimal differentiating feature: (a) thumb on index's distal phalanges (DT), (b) thumb on index's intermediate phalanges (IT), (c) thumb on proximal index phalanges (PT), and (d) thumb and hand in resting position (NT)

We are not the first to bring in deep learning algorithms and computer vision together. For example, previous efforts include robust hand tracking using a depth image(s) [7], [12], [13], [14], [15], [16], using acoustic signals [17], [18], Radio Frequency (RF) based techniques [19], [20], [21], and multi-layered infrared camera arrays [22]. However, these techniques need special sensors or depth cameras in order to function. Use of Kinect like depth cameras limits the mobility during interaction as user has to stand in front of the static camera. Acoustic and RF signal based techniques suffer from signal attenuation and noise issues limiting the active tracking area; moreover, precise finger tracking is difficult to achieve.

Given the current number of smartphone users, we argue that hand gesture tracking should work with off-the-shelf camera. Recently, GoogLeNet [23], a type of deep neural network, has achieved remarkable results on an image classification task across 1000 unique categories. Note that the images used during the training of GoogLetNet were natural images. Hence, we built a similar neural network of smaller size to classify four image categories, possibly, in real-time.

We trained our classifier on synthetically generated dataset and tested against grayscale images. The purpose of testing with grayscale images is to validate the usability with smartphone camera. Our best classifier achieved accuracy of 57%; however, due to the low accuracy we encountered issues to use it in real-time settings. Nonetheless, experimentation has helped us to identify issues with the current approach and provided constructive inputs for future extensions. We believe that using a localization techniques [24] and input data augmentation would improve the accuracy.

Our paper contributes following results:

- Comparative performance analysis of different neural networks against synthetic micro hand gesture dataset. Highlighting parameter tuning and training methods.
- 4-Class micro hand gesture dataset. This dataset consist of labeled depth images across three positive and one negative category. We also discuss the data generation pipeline by using an open-source rendering tool [25] for further research.
- We publish source code to a public repository alongside the dataset. [1].

II. Related Work

Hand gesture detection has received significant attention due to its applications in Human-Computer Interaction, motion control, robotics, and virtual reality. Prior work tends to investigate hand detection from third party perspectives with the intent to recreate the kinematic model of the hand in three dimensional space. In contrast, our work looks at micro hand gestures from an egocentric point of view.

A. Hand Pose Reconstruction

Hand pose reconstruction is a heavily researched field due to its applicability in a wide range of tasks and commercial industries [26], [27], [28], [29]. Furthermore, with the onset of smartphone cameras and sensing devices becoming near ubiquitous, researches are seeking to find a robust solution to hand localization using off-the-shelf products like the Microsoft's Kinect [30] or Intels RealSense cameras [31]. More recently, the advances in hand tracking technologies are beginning to use the state-of-the-art methods in machine learning and deep neural networks. In the paper, "Accurate, Robust, and Felxible Real-time Hand Tracking", Sharp et al. purposed a method to track a hand through a depth sensor as input [7]. The entire pose of the hand is reconstructed though a pipeline that starts with RoI (Region of Interest) extraction, which preceeds to a re-initialization phase that predicts a a distribution over all possible hand poses, and finally ending with a model fitting phase against the produced distributions. In a recent paper, Mueller et al. proposed the use of two convolutions neural networks (CNN) in a pipeline for real time hand pose reconstruction. The two networks are used sequentially, where one estimates the position of the hand within a depth image, also referred as hand localization, and the other uses the estimated hand location to then perform regression on the hands kinematic pose [32].

B. Hand Gesture Detection

The increasing ubiquity of sensing devices like RGB, IR, and depth cameras has fostered a growing interest in using the hand as a control mechinsim for interaction with digital devices. In two papers by Molchanov et al. they look at hand gestures as a novel input mechanism for controlling systems in a vehicle [33], [34]. In their earlier work, they proposed the use of three dimensional CNN to process temporal hand gestures. The architecture consisting of a two branch neural network where one branch process a down-sampled image, and the other a depth image without down-sampling. Later in 2016, they proposed the use of a recurrent neural network (RNN) for the detection of 25 temporal hand gestures.

[1]https://goo.gl/FPJKYR

III. Hand Micro-Gesture

A single hand micro-gesture (SHMG) is composed of diminutive finger movements where the most significant differentiating factor is subtle and may not be obvious to external observers. These type of gestures are suitable for interactions in small space and for accuracy sensitive tasks. Chan et al. [11] performed one of the initial investigations in this area. They conducted a user elicitation study where 16 participant provided different microgestures based on the typical user interface control tasks, for instance, cut, copy, and paste. In another study, Wolf et al. [35] analyzed various microactions within the context of driving a car, using an ATM machine, and a drawing task. They provide a gesture taxonomy based on the context of usage and primary performance trade-offs.

In ideal cases, a classifier that could detect a complete set of microgestures would be very encouraging. However, in this paper, we try to address the part of the problem. In order to test the feasibility of micro-gestures classification of the images fed by a monocular camera, we limit the gesture set only to a thumb and an index finger. Furthermore, our gesture set (see figure 1) is primarily motivated by their potential use within an augmented or virtual reality environment.

We propose a set of microgestures consisting of four hand poses. First, thumb's distal phalanges touching the index finger's distal phalanges (DT); Second, thumb's distal phalanges touching the index's intermediate phalanges (IT); Third, thumb's distal phalanges touching the index's proximal phalanges (PT); and the fourth, where thumb is not touching any finger and the hand is in a resting posture (NT).

For camera positioning we adopt an approach similar to Mueller et al. [32]. We position the frustum of the camera so that its field of view (FOV) constructs an egocentric view of the microgesture. The motivation behind this stems from current and future trends in virtual reality where head-mounted displays (HMD) are increasingly being equiped with sensing technologies. Currently, it is popular for user to mount a camera, such as the Leap Motion, on to the front of a Vive [2] or an Oculus Rift [1]. This approach has the promise of easy and intuitive interaction within virtual environments.

IV. Model Architecture

This section covers the three architectures used during the training and evaluation of our synthetically generated microgesture dataset. The architectures cover both deep and shallow designs. The deepest architecture we used was a variation of the GoogLeNet Inception V3, and the shallowest was a simple CNN with three hidden layers.

A. Transfer Learning

Transfer learning is the task of using one data distribution over a domain and re-purposing it for a data distribution in an another domain [36]. It provides two main advantages – first, it

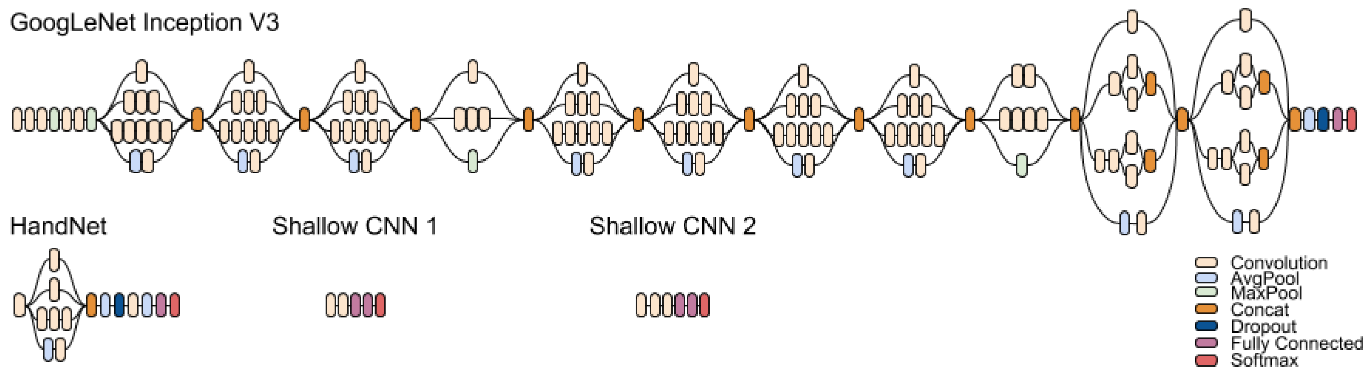

Fig. 2: Set of neural network architectures used during the training and evaluation of the synthetic microgesture dataset. The architectures consist of: (1) a modified version of GoogLeNet Inception v3 where input and output logits have been updated, (2) a custom made deep CNN architecture (HandNet) that emphasizes large spatial features, and (3) simple CNN classifiers

leverages the prior trained network usually trained on massive dataset, in the hope that it will perform better at the similar task with relatively smaller dataset. Usually, a pre-trained network has already learned useful features for a specific task and re-purposing it for a same task with different input dataset has proven to be a viable option. This approach is quite popular and had been used in many state-of-the-art network(s). For instance, Redmon et al. [24] used GoogLeNet Inception for real-time object classification and object localization [24] and Mueller et al. [32] used it *ResNet50* [37] for hand localization task. Common architecture used for transfer learning include AlexNet [38], ResNet [39], and GoogLeNet Inception [40], [41].

As a result, we opted to try out transfer learning as the first step of our experimentation. Specifically, we used aforementioned Inception V3 (see figure 2) network, which was pre-trained over a million natural images and for the similar image classification task as ours. However, we posses relatively small sized dataset (100K images per category) and expect the classifier to work with a very few categories (four categories). The modification to the model reside in the input dimensions and final logit layers. Here a gray scale image $I \in \mathbb{R}^{N\times M\times 1}$ is mapped onto a prediction tensor $P \in \mathbb{R}^{4\times 1}$ before being processed with a softmax layer. All layers in the network had rectified linear units (ReLU) as their activation function, which are defined as:

$$f(z) = max(0, z) \tag{1}$$

Here the variable z represents the input into the acativation function. The layers in our network that are similar to GoogLeNet were initialized with weights from the original GoogLeNet architecture trained on the ImageNet dataset. The entire network was then trained for $40,000$ iterations were the total loss would plateau between 0.24 and 0.25, whcih is depicted in figure (3).

B. Shallow Layers

We constructed two shallow CNN architecture that has similarities to the AlexNet architecture that was original presented by Alex Krizhevsky in 2012 [38]. The first shallow architecture consisted had 2 CNN layers, 2 fully connected layers, and was also followed by a softmax layer for the predictions. The second shallow architecture consisted of three CNN layers, two fully connected (FC) layers, and a final softmax layer. In both the architectures, each of the layers use batch normalization [42] to reduce over-fitting and to speed up the training process. Further, each layer uses exponential linear units (ELU) as their activation function [43], differing from the activation functions used in Inception V3. In contrast to the ReLU activation functions, ELU can maintain a negative value but due to its construction it tends to push the mean of the activation closer to zero. The main differing factoring being the second term in the peicewise function when the input value falls below zero:

$$f(z) = \begin{cases} z & z > 0 \\ \alpha(\exp(z) - 1) & z \leq 0 \end{cases} \tag{2}$$

The hyperparameter α controls the value which an ELU saturates for negative inputs. In the paper, Clevert et al. [43] continue to explain ELU's benefits over other activation functions stating that neural networks using ELU tend to learn quicker and show significant improvements to the networks generalization abilities.

C. HandNet

The final architecture, *HandNet*, has a multi-layer CNN framework that borrows implementation ideas from the inception module proposed by Szegedy et al. [40]. However, there are some notable features that differentiate our model from the Inception V3 architecutre.

The first differentiating feature consist of the activation function (see equation 2). Similar to the activation functions in the simple CNN architecture, HandNet also uses ELU activation on all CNN layers. Second, all the kernel size within the CNN layers are increased, with input CNN layer having an initial kernel dimension of $K \in \mathbb{R}^{21\times 21}$. A motivating factor for the kernel dimension increase is to emphasize the spatiality of the features in synthetic input tensors. Given the

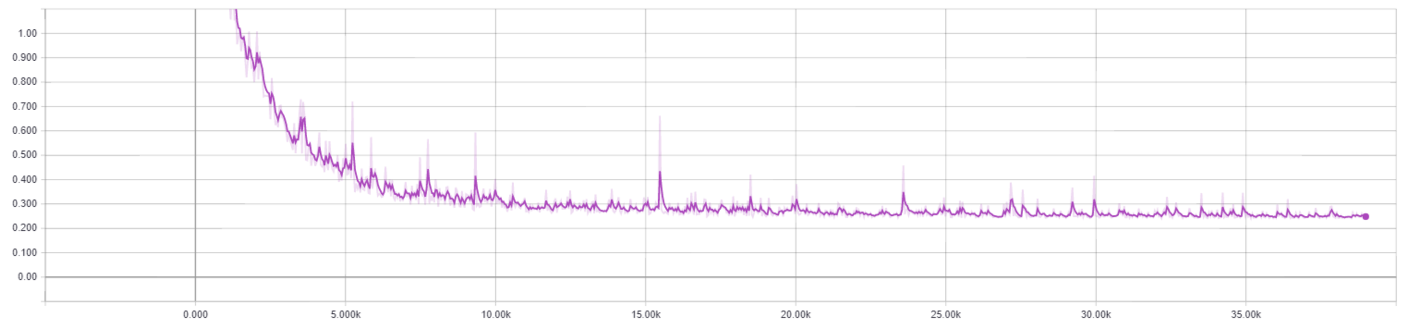

Fig. 3: The total loss calculated the cross entropy on the logits of a modifed GoogLeNet architecture.

nature of our generated dataset, images tend to have large spatial features, and we hypothesis that increasing the kernel size will emphasize these features. Third, the model does not use dimensionality reduction in the inception layer, and instead emphasize kernel depth, were the depth of the kernels per CNN has been increased. The architecture comparisons can be seen in figure (2), and the Tensorflow Slim [44] implemenation can be viewed at Appendix B.

V. Data Generation

There exists a significant number of video and image datasets for the hand as well as for whole body pose estimation [45], [46], [47], [48], [49], [50], [51], but to our knowledge, a dataset for single-handed microgestures is not available. Another problem we encountered was that generating a data from actual participants is a laborious and tedious process. To tackle similar issues, in past, researchers have incorporated synthetic data into the training phase of a neural network. This approach can be seen in FlowNet [52], where the researchers successfully trained an optical flow CNN predictor on synthetically generated flying chairs.

Consequently, we propose the generation of a synthetic single-hand microgesture dataset using a 3D actuated hand model within an open source rendering tool, Blender [25]. With this tool, high-quality hand poses can be rendered along with a high degree of variation by perturbing the 23 degrees of freedom a biological hand provides. Two variations on the our proposed microgesture set were generated. One dataset was generated with a low degree of variation in the positions and angles of each finger. The second dataset was generated with a high degree of variation in all non-index fingers (see figure 9 in appendix). Note that the hand model we used for rendering follows the constraints of a real human hand [53]. With this approach, we hypothesize that the synthetically generated dataset will provide sufficient variability to robustly detect the microgestures in a real-world settings.

A. *Kinematic Model of the Hand*

Given the set of our proposed microgestures (see figure 1), we constructed a hand model that would provide the base for the generated dataset. For each of the interphalangeal joints, we provided constrained degrees of freedom that we empirically determined would fall in our two dataset variations, namely the low and high variation counterparts. To determine the range of perturbation allowed from our hand model, all the constrained ranges are based on the biomechanical properties of the human hand. There has been a significant amount of research into kinemetic hand models that we were able to use in the design of the constrainted ranges. In particular, the work by Cerveri et al. [54], which investigated the construction of a kinematic hand model to use with a marker tracking system like the Vicon [55] and OptiTrack Systems [56]; helped in determining the emperical ranges.

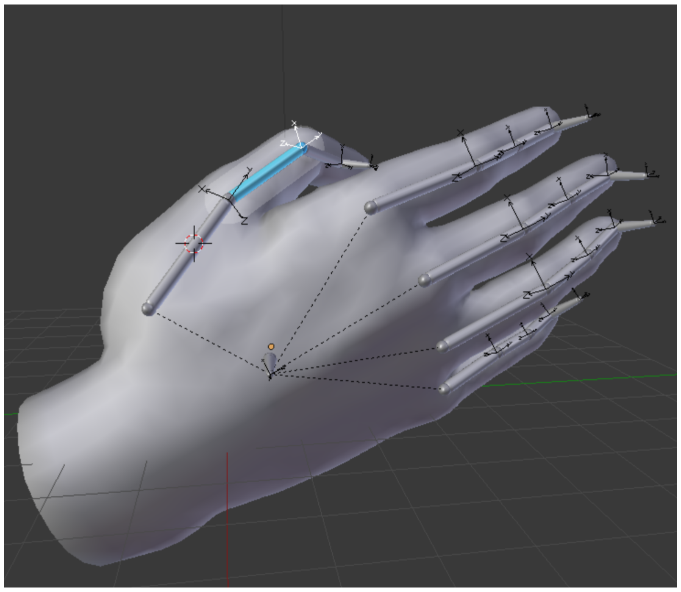

Fig. 4: Hand model used in the geneartion of the micro-posture dataset.

B. *Tools*

With a working skeletal model of a hand (see figure 4), we transformed the hand posture to fit our initial proximal phalanx microgesture (see gesture C in figure 1) as a starting state within the Blender platform. Using the Python language with Blender's integrated APIs, the script renders the various positions of the hand by perturbing the model with varying range of degrees using a uniform distribution that was determined both empirically and through analysis of biomechancial finger properties. This resulted in a high degree of subtle variation across both datasets. The low variation dataset restricted the degree of perturbation for the middle, ring, and little finger to be within $+5$ to -5 degrees of variability. The high variation dataset used a restricted range between $+15$ to -30 degrees of variability when rendering the set for each middle, ring and little finger's metacarpophalangeal joints. In total, each dataset took approximately 8 hours to generate.

C. Dataset

A total of 2 datasets were generated with a total of $400,000$ images each, $100,000$ for each category. The total size across both dataset totaled more than 24 GB of data. For testing against real data, a small dataset of 1000 images were generated using the Leap Motion for each of the four categories. A comparison of the real and generated data can be viewed in figure (8).

VI. Method

This section describes the method and pipeline used to train and evaluate the various models.

A. Data Augmentation

The synthetic dataset consist of $400,000$ images where each category contains one quarter of the total. To prevent over-fitting to the datasets underlying probability distribution, we performed online data augmentation on the tensor representation of the image before it was passed into the model. The prepossessing consisted of random cropping, image resizing, random inversion across the x-axis (horizontal), colour distortion, and normalization.

In the first phase, the image tensor is randomly cropped so that its total new area consists of approximately 87% of its total area. The image is then resized to a dimension of $I \in \mathbb{R}^{299\times299\times3}$ which then is subsequently passed into a randomized method where the image has a 50% chance of inverting across its x-axis.

The second phase of augmentation results in image distortion over the colour channels. These channels are perturbed with varying degrees of strength on four different metrics: (1) brightness, (2) saturation, (3) hue, and (4) contrast. Considering that each colour distortion is non-communicative, the ordering of these operation matter for the final distorted image. Thus, the ordering of the distortion is chosen randomly at runtime to increase the possible variation of the resulting distorted image. The image is then converted to gray scale with the final tensor being $I \in \mathbb{R}^{299\times299\times1}$.

The final stage of augmentation results in the image being normalized between $[-1,+1]$ and the mean of the image subtracted from the original. The normalized image is added to the batch of images, in the case of HandNet the batch size is 16, and the final tensor passed to the model is $I \in \mathbb{R}^{16\times299\times299\times1}$

B. Training

The process of training each CNN architecture consists of updating the weights W of the model with respect to some cost. In the case for all four networks the cost function, $L(\mathcal{W},\mathcal{D})$, used to calculate the loss across the predictions is the standard softmax cross entropy. The softmax layer estimates the class probability based on the given input tensor,

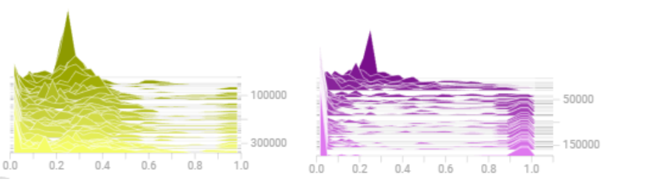

Fig. 5: Histogram of the model's logits activations from the softmax layer for both HandNet trained on high variation data (left) and low variation data (right). The x-axis represents the value presented in the weights. The y-axis represents the number of weights containing the value on the x-axis. The z-axis represents the step number.

$P(C|x,\mathcal{W})$. The optimizer used for weight updates is an implementation of the ADAM algorithm proposed by Kingma et al. in 2015 [57]. The ADAM optimizer uses both the first and second order moment and decays them over time as the global step size increases. In the paper, the authors demonstrate its superiority in dealing with large parameter spaces and in its ease of hyperparameter tuning as compared with other optimizers like RMSProp [58] and AdaGrad [59].

All networks were trained until there loss estimate converged to some horizontal asymptotic threshold, which usually fell in the range $[0,+1]$.

C. Evaluation

During the evaluation process, each model was validated on both a subset of the synthetic data as well as a small sample of real images. Metrics were collected for the global accuracy with respect to all classes and the accuracy for each individual class.

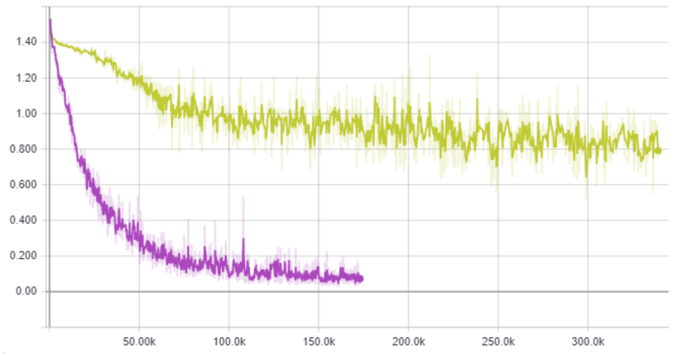

Fig. 6: Total loss (y-axis) per global step count (x-axis) for the HandNet model trained on high variation data (yellow), and low variation data (purple).

VII. Experiments

To properly evaluate our models, a series of test were run to determine the accuracy each model had on our purposed gesture set in a real world setting. The initial test was against a validation set of gestures which was a subset of our synthetically generated data. This always resulted in the model achieving a global accuracy between 95% to 99% for each model. The second test was against the real data generated

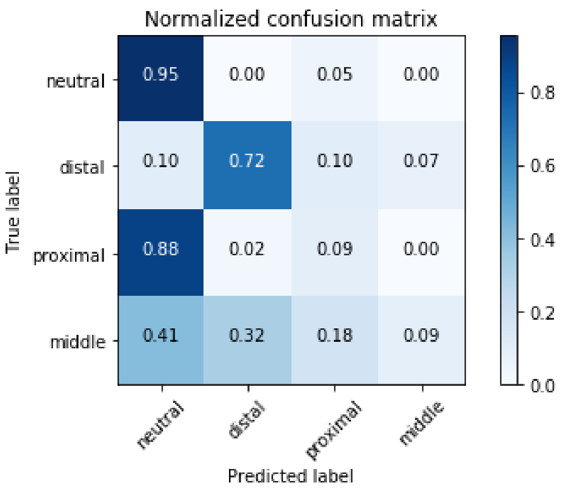

Fig. 7: Confusion matrix for a sub-sample of the real training images processed on the low variation HandNet model.

from a Leap Motion sensor which had far more interesting and varied results.

When analyzing each model during the training process, the generated histogram for the logits activations proved to be a useful tool and provided precise insight into the underlying distributions within the model's layers.

The HandNet model was trained on two variations of the synthetic dataset (See appendix A). During training the total loss from the softmax cross entropy cost function and the activations on the logits become clear differentiators that separated the model between the two dataset variations. In figure (5), the activations of the logits can be viewed within the softmax layer. As the model progresses through time (z-axis), the actual predictions the model makes should become more confident resulting in more activations registered near zero and one. This observation can be seen in the histogram on the right (in purple) for the HandNet model trained on the low variation dataset for $150,000$ global steps. Here the model is making bolder predictions for each class and the resulting gap in the middle is a result of these predictions as more activations are recorded closer to zero and one. When running the validation test, the model achieved a 98.2% accuracy. In contrast to this, the HandNet model trained on the high variation dataset (left) for $300,000$ global steps makes less bold predictions and tends to accumulate its activations within the range [0.0 - 0.5]. Unsurprising the resulting accuracy on the validation set is 62.3%.

Another telling indicator is the history of the total loss with respect to the global time step. There appears to be a direct correlation between the loss and the activations within the logit layers. Using the same two models as above, the resulting loss for each can be seen in figure (6). The model trained on the low variation data (purple) converged much sooner and has a relatively smoother curve then the model trained on the high variation set (yellow).

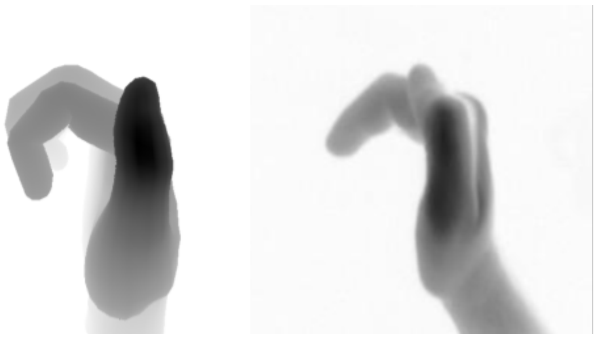

Fig. 8: Comparison of the synthetically generated data (left) with the captured real image data from the Leap Motion (right).

A. *Results*

The summary of our results can be viewed in Table I, which displays the average global accuracy and the accuracy broken down by gesture type across each of neural network models.

We hypothesis that the low score seen across all the networks when running the model against the real image dataset is due to the discrepancies between the synthetic and real data. The synthetic data happens to be generated in a controlled environment, where the depth values from the camera to the model can be exactly computed. In the real image data, the image itself is exposed to noise from the environment and the actual image is taken with an infrared (IR) camera which will not exactly reflect the data distribution of the synthetic dataset. A comparison of the synthetic data and real image data can be seen in figure (8).

One interesting result form the network architecture comparison is that the average accuracy of the HandNet model exceeding that of the much larger Inception V3 model. A possible explanation for this result could be combination of the ELU activation functions, the use of larger convolution kernels, and by preventing dimensionality reduction on the CNN branches within HandNet as is explained in section IV-C. Our highest rated CNN classifier is Shallow CNN 2, whose architecture can be viewed in figure (2). One possible explanation as to why this shallow CNN outperforms the deeper networks could be due to the input dimension of the network. As compared with the other networks, which take an input tensor of $I \in \mathbb{R}^{16\times299\times299\times1}$, this network reduced the size of the image to 100×100, leaving the input tensor as $I \in \mathbb{R}^{32\times100\times100\times1}$. Similar to our previous claims, we hypothesis that this improvement is due to the emphasis smaller images have on spatially large features. We leave a more thorough analysis of this result to future work.

The confusion matrix in figure (7) shows the predicted probabilities for each category as compared with thier ground truth labels. The matrix was generated by processing HandNet with a random sub-sampling of the real dataset captured through the Leap Motion sensor with sample size $N = 256$. Interestingly and somewhat unsurprisingly, the model appears to be able better distinguish between neutral and distal microgestures but miss-classify the proximal and middle microgestures. We hypothesis that this result is due to the minimal differentiating features between the proxmal and

Network Architecture	Average Accuracy	(a)	(b)	(c)	(d)
Shallow CNN 1	29.16%	20.83%	37.50%	20.83%	29.16%
Shallow CNN 2	**57.43%**	**80.50%**	74.80%	40.00%	52.50%
HandNet	44.14%	31.25%	45.83%	39.39%	37.50%
DNN (Inception V3)	34.50%	37.50%	43.75%	39.58%	35.41%

TABLE I: **Comparison of network models:** The table highlights the difference between the various networks trained on the synthetically generated data and tested on real images.

middle microgestures as compared with the neutral and distal categories.

VIII. Future Work

Experimenting with different variations of architectures helped us understand the training and testing process more thoroughly as well as to point out the pitfalls of our current methodology. Based on empirical observations, we speculate that our models can be improved in numerous ways. Some of the possibilities are discussed below.

A. Hand Localization

Each model used during the experimentation process was tested against a mixed dataset. As mentioned before, this dataset combines a major chunk of synthetic and a small chunk of grayscale images (collected from a monocular camera). The problem we speculate here is that the hand location can span within the image resolution (500×500) and it might cause issues to accurately classify the gesture. Due to the variation in hand size, distance from camera and illumination. On the other hand, if we use a hand localization technique to filter out the unnecessary details from the image, it might improve the recognition accuracy. A possible solution would be to extend upon the approaches discussed in [24], [60], [61]. Hence, we plan to incorporate hand localization to improve the accuracy.

B. Personalized Classifier

Generating synthetic data saves a significant amount of time as compared to outsourcing labeling tasks with human computation platforms like Amazons Mechanical Turk [62]. Furthermore, it allows researcher to make quick variations and provides reliable means for a very accurate ground truth. However, a classifier trained on specific individual characteristics would allow it to more efficiently generalize to that individuals traits. Consider following scenario – A smartphone user can record a short video for each microgesture category, which then can be fragmented into training images to feed into the network. We speculate that the model trained on the dataset generated by an individual would work with higher accuracy. A caveat, however, to such approach is that it would not scale across different users. Recent advances in one-shot learning [63] could be used to provide higher accuracy for individual users.

C. Sequence Pooling

Our models classify individual images, and lack knowledge of their sequential structure. In other words, if a single frame of a video stream is classified as a class-1 then the next frame would have high likelihood to belong to same class as the previous frame which is the apprach taken by Malchanov et al. [33] Hence, rather than training on individual images, training on a short video clips would improve the robustness as well as the accuracy of a classifier as it leverages correlations in the underlying data distribution that take place over time. Moreover, microgestures which have temporal components associate with it, for example, swipe left / swipe up etc., would fit in the model.

Albeit this approach is suitable for temporal microgestures, it might introduce latency during the recognition. Classifier need a fixed set of images to make a decision as well as segment the video, however, investigation of a real-time performance is still open for exploration.

IX. Conclusion

In this paper, we propose the first microgesture classification technique using deep neural networks. Our efforts included a comparative analysis of four different types of networks and generating a synthetic dataset from scratch. Unlike the previous efforts of hand classification, which primarily work with RGB-D images, we could achieve the accuracy up to 57% against the images produced using a monocular camera. This hints the possibility of using a classifier in ubiquitous environments, especially, where users have the smartphone camera. However, accuracy of the current classifier can further be improved with additional experiments. To encourage research in this direction we are releasing the pre-trained network and the dataset used in our experiment to the public.

References

[1] “Oculus Rift,” https://www.oculus.com/rift/, accessed: 2010-08-12.

[2] “HTC Vive,” https://developer.leapmotion.com/vr-setup/vive, accessed: 2017-08-12.

[3] M. Weigel, T. Lu, G. Bailly, A. Oulasvirta, C. Majidi, and J. Steimle, “iSkin: Flexible, Stretchable and Visually Customizable On-Body Touch Sensors for Mobile Computing,” in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI ’15*. New York, New York, USA: ACM Press, 2015, pp. 2991–3000.

[4] M. Weigel, A. S. Nittala, A. Olwal, and J. Steimle, “SkinMarks: Enabling Interactions on Body Landmarks Using Conformal Skin Electronics,” *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI ’17*, pp. 3095–3105, 2017. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3025453.3025704

[5] D. A. Bowman, C. Wingrave, J. Campbell, and V. Ly, "Using pinch gloves (TM) for both natural and abstract interaction techniques in virtual environments," *HCI International*, no. 0106, pp. 629–633, 2001. [Online]. Available: http://eprints.cs.vt.edu/archive/00000547/

[6] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, p. 116, jan 2013.

[7] T. Sharp, Y. Wei, D. Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, S. Izadi, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, and A. Vinnikov, "Accurate, Robust, and Flexible Real-time Hand Tracking," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*. New York, New York, USA: ACM Press, 2015, pp. 3633–3642.

[8] R. Walter, G. Bailly, N. Valkanova, and J. Müller, "Cuenesics: Using Mid-Air Gestures to Select Items on Interactive Public Displays," *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services - MobileHCI '14*, pp. 299–308, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2628363.2628368

[9] R. Walter, G. Bailly, and J. Müller, "StrikeAPose: revealing mid-air gestures on public displays," *SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*, pp. 841–850, 2013. [Online]. Available: http://joergmueller.info/pdf/CHI13WalterStrikeapose.pdf

[10] M. Nancel, J. Wagner, E. Pietriga, O. Chapuis, and W. Mackay, "Mid-air Pan-and-Zoom on Wall-sized Displays," pp. 177–186, 2011.

[11] E. Chan, T. Seyed, W. Stuerzlinger, X.-D. Yang, and F. Maurer, "User Elicitation on Single-hand Microgestures," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*. New York, New York, USA: ACM Press, 2016, pp. 3403–3414.

[12] S. Sridhar, F. Mueller, A. Oulasvirta, and C. Theobalt, "Fast and Robust Hand Tracking Using Detection-Guided Optimization."

[13] J. Sanchez-Riera, K. Srinivasan, K.-L. Hua, W.-H. Cheng, M. A. Hossain, and M. F. Alhamid, "Robust rgb-d hand tracking using deep learning priors," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[14] C. Wan, T. Probst, L. Van Gool, and A. Yao, "Crossing nets: Combining gans and vaes with a shared latent space for hand pose estimation."

[15] E. Krupka, K. Karmon, N. Bloom, D. Freedman, I. Gurvich, A. Hurvitz, I. Leichter, Y. Smolin, Y. Tzairi, A. Vinnikov *et al.*, "Toward realistic hands gesture interface: Keeping it simple for developers and machines," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2017, pp. 1887–1898.

[16] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1106–1113.

[17] C. Zhang, Q. Xue, A. Waghmare, S. Jain, Y. Pu, J. Conant, S. Hersek, K. A. Cunefare, O. T. Inan, and G. D. Abowd, "Soundtrak: Continuous 3d tracking of a finger using active acoustics," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 2, p. 30, 2017.

[18] W. Wang, A. X. Liu, and K. Sun, "Device-free gesture tracking using acoustic signals," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, 2016, pp. 82–94.

[19] J. Lien, N. Gillian, M. E. Karagozler, P. Amihood, C. Schwesig, E. Olson, H. Raja, and I. Poupyrev, "Soli: Ubiquitous gesture sensing with millimeter wave radar," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 142, 2016.

[20] S. Wang, J. Song, J. Lien, I. Poupyrev, and O. Hilliges, "Interacting with soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, 2016, pp. 851–860.

[21] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota, "Fingerio: Using active sonar for fine-grained finger tracking," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2016, pp. 1515–1525.

[22] Y. Takeoka, T. Miyaki, and J. Rekimoto, "Z-touch: a multi-touch system for detecting spatial gestures near the tabletop," in *ACM SIGGRAPH 2010 Talks*. ACM, 2010, p. 57.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[25] "Blender," https://www.blender.org/, accessed: 2010-08-12.

[26] H. Guan and M. Turk, *3D Hand Pose Reconstruction with ISOSOM*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 630–635. [Online]. Available: https://doi.org/10.1007/11595755_77

[27] M. Bianchi, P. Salaris, and A. Bicchi, "Synergy-based hand pose sensing: Reconstruction enhancement," *CoRR*, vol. abs/1206.0555, 2012. [Online]. Available: http://arxiv.org/abs/1206.0555

[28] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun, *Real Time Hand Pose Estimation Using Depth Sensors*. London: Springer London, 2013, pp. 119–137. [Online]. Available: https://doi.org/10.1007/978-1-4471-4640-7_7

[29] V. Athitsos and S. Sclaroff, "Estimating 3d hand pose from a cluttered image," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2, June 2003, pp. II–432–9 vol.2.

[30] "Kinect for xbox one | xbox," (Accessed on 08/09/2017).

[31] "IntelÂő realsenseâĎć technology," https://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html, (Accessed on 08/09/2017).

[32] F. Mueller, D. Mehta, O. Sotnychenko, S. Sridhar, D. Casas, and C. Theobalt, "Real-time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor," p. 8, 2017.

[33] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand gesture recognition with 3D convolutional neural networks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, jun 2015, pp. 1–7.

[34] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, "Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016, pp. 4207–4215.

[35] K. Wolf, A. Naumann, M. Rohs, and J. Müller, "LNCS 6946 - A Taxonomy of Microinteractions: Defining Microgestures Based on Ergonomic and Scenario-Dependent Requirements," *LNCS*, vol. 6946, pp. 559–575, 2011.

[36] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, oct 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5288526/

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances In Neural Information Processing Systems*, pp. 1–9, 2012.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

[41] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," *Arxiv*, vol. 42, p. 12, feb 2016.

[42] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html

[43] D. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *CoRR*, vol. abs/1511.07289, 2015. [Online]. Available: http://arxiv.org/abs/1511.07289

[44] “models/readme.md at master Âů tensorflow/models,” https://github.com/tensorflow/models/blob/master/inception/inception/slim/README.md, (Accessed on 08/13/2017).

[45] “Multi-modal gesture recognition,” https://www.kaggle.com/c/multi-modal-gesture-recognition/data, accessed: 2010-08-12.

[46] “3D Hand Gesture Recognition,” http://www-rech.telecom-lille.fr/shrec2017-hand/, accessed: 2010-08-12.

[47] “MSR Action Dataset,” http://www.uow.edu.au/~wanqing/#MSRAction3DDatasets, accessed: 2010-08-12.

[48] L. Xia, C. Chen, and J. Aggarwal, “View invariant human action recognition using histograms of 3d joints,” in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. IEEE, 2012, pp. 20–27.

[49] J. Zhang, W. Li, P. O. Ogunbona, P. Wang, and C. Tang, “Rgb-d-based action recognition datasets: A survey,” *Pattern Recognition*, vol. 60, pp. 86–105, 2016.

[50] A. Memo and P. Zanuttigh, “Head-mounted gesture controlled interface for human-computer interaction,” *Multimedia Tools and Applications*, pp. 1–27, 2016.

[51] G. Marin, F. Dominio, and P. Zanuttigh, “Hand gesture recognition with leap motion and kinect devices,” in *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1565–1569.

[52] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, “FlowNet: Learning Optical Flow with Convolutional Networks,” in *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, dec 2015, pp. 2758–2766.

[53] S. Cobos, M. Ferre, M. S. Uran, J. Ortego, and C. Pena, “Efficient human hand kinematics for manipulation tasks,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 2246–2251.

[54] P. Cerveri, N. Lopomo, A. Pedotti, and G. Ferrigno, “Derivation of Centers and Axes of Rotation for Wrist and Fingers in a Hand Kinematic Model: Methods and Reliability Results,” *Annals of Biomedical Engineering*, vol. 33, no. 3, pp. 402–412, jan 2005.

[55] “Motion capture systems | vicon,” https://www.vicon.com/, (Accessed on 08/09/2017).

[56] “Optitrack - motion capture systems,” http://optitrack.com/, (Accessed on 08/09/2017).

[57] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pp. 156–160, dec 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[58] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[59] J. Duchi, E. Hazan, and Y. Singer, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,” *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[60] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, “Efficient object localization using convolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 648–656.

[61] W. Yang, S. Li, W. Ouyang, H. Li, and X. Wang, “Learning feature pyramids for human pose estimation,” *arXiv preprint arXiv:1708.01101*, 2017.

[62] “Amazon mechanical turk - all hits,” https://www.mturk.com/, (Accessed on 08/13/2017).

[63] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. P. Lillicrap, “One-shot learning with memory-augmented neural networks,” *CoRR*, vol. abs/1605.06065, 2016. [Online]. Available: http://arxiv.org/abs/1605.06065

Appendix A
Comparison of the high and low variation dataset across all categories.

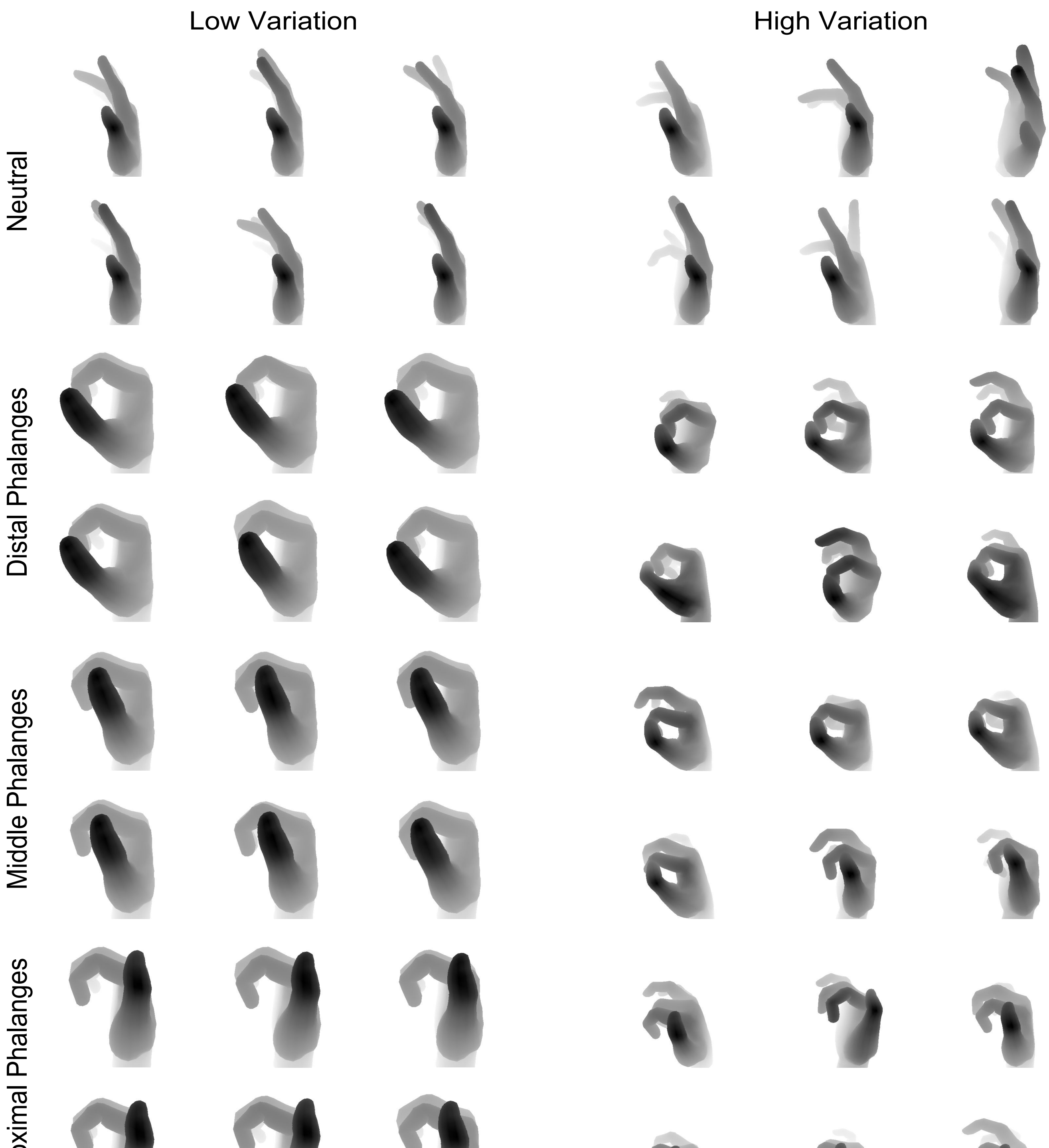

Fig. 9: Synthetic dataset generated using a 3D model of the hand with a kinematic model to ensure accurate hand posture. The dataset consist of four categories: (1) Neutral, (2) Distal Phalanges, (3) Middle Phalanges, and (4) Proximal Phalanges. Each set of the four categories are rendered with a small amount of variation (left) and with a large amount of variation (right).

APPENDIX B
TENSORFLOW SLIM IMPLEMENTATION OF HANDNET.

```
with tf.variable_scope(scope, 'handnet_v2', [inputs]):
    with slim.arg_scope([slim.conv2d, slim.max_pool2d, slim.avg_pool2d],
                        stride=1, padding='VALID'):
        # 299x299x1
        end_point = 'Conv2d_1_21x21'
        net = slim.conv2d(inputs, 64, [21, 21], stride=2, scope=end_point)
        end_points[end_point] = net
        # 139x139x64

    # handnet v2 block
    with slim.arg_scope([slim.conv2d, slim.max_pool2d, slim.avg_pool2d],
                        stride=2, padding='SAME'):
        with slim.arg_scope([slim.conv2d], activation_fn=tf.nn.elu):
            end_point = 'Mixed_Section_1'
            with tf.variable_scope(end_point):
                with tf.variable_scope('Branch_0'):
                    branch_0 = slim.conv2d(net, 128, [3, 3], scope='Conv2d_0a_3x3')
                with tf.variable_scope('Branch_1'):
                    branch_1 = slim.conv2d(net, 64, [1,1], scope='Conv2d_0a_1x1')
                    branch_1 = slim.conv2d(branch_1, 96, [3,3], stride=1, scope='Conv2d_0b_3x3')
                    branch_1 = slim.conv2d(branch_1, 96, [3,3], stride=1, scope='Conv2d_0c_3x3')
                with tf.variable_scope('Branch_2'):
                    branch_2 = slim.conv2d(net, 32, [11, 11], scope='Conv2d_0c_11x11')
                with tf.variable_scope('Branch_3'):
                    branch_3 = slim.avg_pool2d(net, [3,3], stride=1, scope='AvgPool_0a_1x1')
                    branch_3 = slim.conv2d(branch_3, 96, [5, 5], scope='Conv2d_0d_5x5')

        net = tf.concat(axis=3, values=[branch_0, branch_1, branch_2, branch_3])
        end_points[end_point] = net

    # Prediction and logits
    with slim.arg_scope([slim.batch_norm, slim.dropout],
                        is_training=is_training):
        with slim.arg_scope([slim.conv2d], padding='VALID', activation_fn=tf.nn.elu):
            with tf.variable_scope('Logits'):
                net = slim.avg_pool2d(net, [8, 8], scope='AvgPool_1a_8x8')
                net = slim.dropout(net, keep_prob=dropout_keep_prob, scope='Dropout_1b')
                net = slim.conv2d(net, 128, [7, 7], stride=2, scope='Conv2d_Final_5x5_1a')
                net = slim.avg_pool2d(net, [8, 8], scope='AvgPool_1b_8x8')
                net = slim.conv2d(net, 256, [3, 3], stride=2, scope='Conv2d_Final_3x3_1b')
                # net = slim.conv2d(net, 512, [3, 3], stride=2, scope='Conv2d_Final_3x3_1c')
                end_points['PreLogits'] = net

                logits = slim.conv2d(net, num_classes, [1, 1], activation_fn=None,
                                     normalizer_fn=None, scope='Conv2d_Final_1x1_1c')
                if spatial_squeeze:
                    logits = tf.squeeze(logits, [1, 2], name='spatialSqueeze')

            end_points['Logits'] = logits
            end_points['Predictions'] = prediction_fn(logits, scope='Predictions')
```

Fig. 10: Implementation of the HandNet model using the Tensorflow Slim APIs. Notable features include the use of ELU activation functions, large kernel sizes, and deep kernel demensionality.