Deep Sparse Autoencoders for American Sign Language Recognition Using Depth Images

Pablo Rivas[1], **Ezequiel Rivas**[2], **Omar Velarde**[2], **and Samuel Gonzalez**[2]
[1]Department of Computer Science, Marist College, New York, USA
[2]Division of Graduate Studies, Nogales Institute of Technology, Sonora, Mexico

Abstract— *In this paper we address the problem of hand gestures recognition in the American Sign Language using machine learning. We propose a five layer unsupervised encoder-decoder neural model. We use a dataset of segmented images captured with a depth-sensor camera for different subjects. The average accuracy obtained was of 98.9% and as high as 99.4% on unseen data.*

Keywords: deep learning, autoencoders, neural networks, sign language, depth imaging

1. Introduction

Language is an essential part of being human as it enables us to communicate with others. Some people have to develop additional means of communication for different reasons. A popular alternative language is through signs. This type of language is based on hand-produced symbols or figures.

Sign language recognition is a task that has been studied in the last few years [1]. We believe this is an important problem to address because it directly affects the quality of life of people who need to communicate using sign language. The task is to develop algorithms that can recognize different signs in an alphabet and that can do so in an efficient manner. Ideally, the methodologies should be fast to the point of enabling the near real-time processing of signs [2], [3].

In this research we deal with the recognition of signs over the American sign language (ASL). The proposed approach uses depth images of subjects making different signs, building upon the work of B. Kang, et.al. in [4].

Typical approaches addressing similar problems involve the usage of hidden Markov models [2], and a combination of them with other discriminative functions for feature extraction in multi-stage architectures [5]. Other major alternatives included the exploration of neural strategies combined with fuzzy systems [6]. For a more detailed review of alternatives for generic hand gesture recognition we can turn to the work in [7].

However, with the dramatic attention that deep learning has gained recently in the machine learning and the image processing for pattern recognition communities [8], [9], we are motivated to similarly explore this new algorithmic alternative to see if it offers better solutions to open problems. Most recently, the authors in [4] have explored a deep learning approach based on convolutional neural networks (CNNs) achieving outstanding results in solving the problem that we address here. Nonetheless, the training of a CNN and its deployment may be computationally expensive and inconsistent [10]; furthermore, other simpler and less costly deep learning alternatives tend to be overlooked [11]. The research presented aims to show that a simpler deep learning approach based on stacked autoencoders in a neural network architecture is capable of solving the same problem with comparable results. We will show that this simple alternative approach also achieves great performance and is, in nature, more efficient [12].

The rest of the paper is organized as follows, Section 2 discusses the state-of-the art in machine learning approaches to classify hand gestures. Section 3 discusses the dataset and proposed architecture, while the experimental setup us addressed in Section 4 along with a discussion of the results. Finally, conclusions are drawn in Section 5.

2. Background and Related Work

Recently, J. Nagi et.al. [13] studied a deep neural network approach for recognizing six different hand gestures given to a robot. Using CNNs the authors achieve a 96% accuracy rate. Then, M. Van den Bergh et.al. [14] explored the idea of using depth-sensor imagery to establish an algorithm for hand segmentation. The authors ultimately used this to recognize six different hand gestures in 3D. They achieved a 99.07% recognition rate. Some of the key components of the overall architecture were the usage of Haar wavelets for image filtering in a classifying methodology known as Average Neighborhood Margin Maximization (ANMM). Van de Bergh's work is similar to an earlier work by Isaacs and Foo in 2004 [15], where the authors use different wavelets to extract features and later feed these features to a feed-forward neural network claiming to achieve a 99.90% recognition accuracy rate, but there is no evidence to support this result.

N. Pugeault et.al. also studied this issue [16]; they combined both segmented color images and depth images processed with Gabor filters [17] as a means to extract features. They classified the processed images using the Random Forests (RF) learning algorithm. The authors report achieving a 69.0% recognition rate using only depth images and a 75.0% using both depth and color image data.

Later, A. Kuznetsova et.al. [18] makes use of 3D modeling to represent depth images and then extract features using such 3D models. These features are then used to train a multi-layered random forest (MLRF). The authors report that their highest recognition rate was 87.0% using half of the data for training and the other half for testing with samples chosen at random.

In 2015, Dong et.al. [19] used depth images and determined the position of each hand joints; they used the joint information to calculate the angle of the joints and used this as features. The authors used an RF classifier and reported accuracies of up to 90%.

Moreover, B. Kang et.al. [4] took advantage of the recent success of CNNs for pattern recognition over images. They propose a system that makes use of depth-sensor images to study further the problem of hand gesture recognition. When recognizing signs within the same subject they averaged a 99.99% accuracy rate over five subjects. However, when recognizing signs across different subjects, their accuracy rate dropped to 83.58%.

Our research aims to further study the problem posed in [4] of increasing accuracy rates over different subjects. In the next section we discuss the dataset used and propose a deep learning approach using autoencoders and a neural network.

3. Methodology

3.1 Data

This research focuses in the ASL as a case study [20]. The ASL has 26 hand gestures corresponding to the letters in the English alphabet; it also contains 9 hand gestures for every single number. Figure 1 depicts the gestures pertaining to our study.

We use the data provided in [4] that was released in 2015. This dataset consists of images acquired with a three-dimensional depth-sensor camera. There is a total of $31,000$ images available with a resolution of 256×256. The images correspond to already segmented dept-images that contain the area of the hand. The authors use a rather simple algorithm to make the segmentation based on the fact that in all depth images the hand is the closest object with respect to the camera.

Not all the 35 signs (26 letters and 9 numbers) were considered in this study. Some signs were excluded and others considered jointly as follows.

The signs corresponding to J and Z clearly require a sequence of images rather than a single instance, as can be seen in Figure 1. This goes beyond the scope of this project; although the detection of signs from live-video sequences is part of a bigger project, we will not address it in this paper. For that reason these two signs were excluded.

The signs corresponding to V and 2, by observation, are nearly identical and their distinction requires contextual information. Similarly, the signs for letter W and number 6 require being interpreted in context. For this reason, these signs were considered as the same sign, for a total of 31.

There is a total of five different subjects and each subject produced 200 images of each sign. This represents a total 1,000 images per sign and 31,000 images overall. The images collected capture each subject making a hand gesture moving across the horizontal axis and varying inclination of the hand gesture, as depicted in Figure 2.

3.2 Sparse Autoencoders

A sparse autoencoder is usually categorized as a neural network with unsupervised learning [21]. In a general sense an autoencoder is trained to output an approximation of the input provided a deep architecture of layered neurons that encode and decode based on the input stimuli. In our research we use a sparse autoencoder that specifically minimizes a modified loss function based on the mean squared error.

Let $\mathbf{x} \in \mathbb{R}^d$ be a d-dimensional input vector. Then the loss function of the common sparse autoencoder is defined as follows:

$$\frac{1}{N}\left\|\mathbf{x}_n - \hat{\mathbf{x}}_n\right\|_2^2 + \theta_w \frac{1}{2}\sum_{l=1}^{L}\left\|\mathbf{w}^l\right\|_2^2 + \theta_s \sum_{m=1}^{M} KL\left(\theta_\alpha \,\|\bar{\alpha}_m\right) \tag{1}$$

where N is the total number of training samples; $\hat{\mathbf{x}}_n$ is the learned (or encoded) approximation of the n-th input vector $\mathbf{x}_n$; θ_w controls the sparseness of the weights of the network, $\mathbf{w} \in \mathbb{R}^d$; L denotes the number of layers in the deep network; θ_s regulates the sparsity of the activation functions' output, α, in every neuron in the network; M is the total number of neurons in the deep network; and $KL(\cdot)$ is the Kullback-Leibler divergence function [22] used to measure how much the observed average activation of the m-th neuron, $\bar{\alpha}_m$, actually deviates from the desired average output, θ_α.

The Kullback-Leibler divergence function can be defined as follows [23]:

$$\sum_{m=1}^{M} KL\left(\theta_\alpha \,\|\bar{\alpha}_m\right) = \sum_{m=1}^{M} \theta_\alpha \log\left(\frac{\theta_\alpha}{\bar{\alpha}_m}\right) + (1-\theta_\alpha)\log\left(\frac{1-\theta_\alpha}{1-\bar{\alpha}_m}\right) \tag{2}$$

where the average output of the m-th neuron, $\bar{\alpha}_m$, at the l-th layer is given by

$$\bar{\alpha}_m = \frac{1}{N}\sum_{n=1}^{N} \psi\left(\mathbf{w}_m^{(l)T}\mathbf{x}_n + b_m^{(l)}\right) \tag{3}$$

where $\psi(\cdot)$ is the neuron's activation function, and $b_m^{(l)}$ is the bias term for the m-th neuron at the l-th layer. In this research we specifically use logistic (sigmoid) activation functions, $\psi(z) = 1/(1-e^{-z})$, for any given value of z.

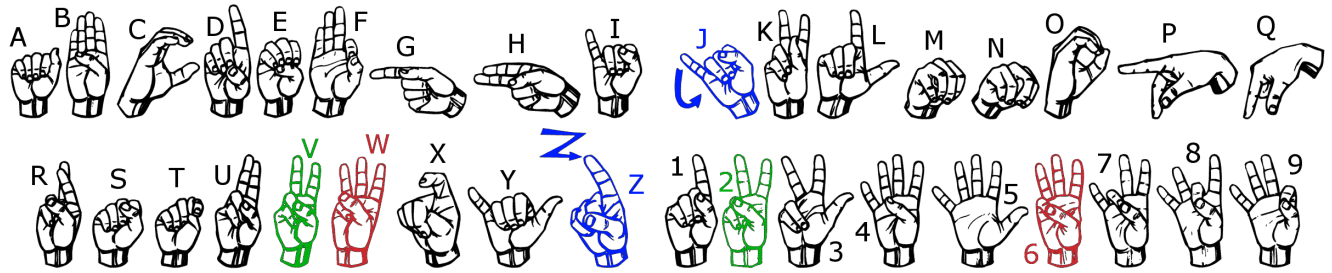

Fig. 1: The American Sign Language (ASL) displaying 26 signs corresponding to the English alphabet (A to Z) and 9 signs for numbers (1 to 9).

Fig. 2: Examples of ASL hand gestures corresponding to the number one and number two with variations in the horizontal axis and its inclination with respect to a depth-sensor camera.

By observing the mathematical form of (1), it is feasible to apply an accelerated form of guided learning known as scaled conjugate gradient (SCG) descent [24]. SCG has proven to be a reliable and efficient form of minimizing loss functions such as the one for the sparse autoencoder, overcoming the shortcomings of a traditional conjugate gradient and a back-propagation with gradient descents [21].

3.3 Deep Learning Architecture

Two stacked autoencoders are combined with a feedforward neural network in a five-layer architecture, as shown in Figure 3. The first two layers are a set of unsupervised autoencoders that minimize the loss function in (1). The first layer, i.e., an encoding layer, receives as input N images of 256×256 as row vectors, each denoted as $\mathbf{x}_n \in \mathbb{R}^{65536}$, where $n \in \{1, 2, \ldots, N\}$. The training phase encodes the attributes using 100 neural units to produce $\hat{\mathbf{x}}_n \in \mathbb{R}^{100}$, and decodes back to the feature space using, intuitively, 65536 neural units; all neural units use logistic activation functions.

Similarly, the third and fourth layers are an encoder and decoder, respectively. The encoder in the third layer receives as input an encoded version of the input coming from the first layer, denoted as $\hat{\mathbf{x}}_n$, and encodes using 50 neural units producing a modified version of the feature vector denoted as $\tilde{\mathbf{x}}_i \in \mathbb{R}^{50}$. The decoder in the fourth layer decodes using 100 neural units.

In the last layer of the model we use a network of 31 neural units with softmax activation functions. Each neuron is stimulated $\tilde{\mathbf{x}}_n$ and is trained to predict the probability of the n-th sample belonging to a specific class $C \in \{1, 2, \ldots, 31\}$. The output of this layer for the n-th sample is the estimated probability of that sample belonging to all classes, formally denoted as $\hat{\mathbf{d}}_n \in \mathbb{R}^{31}$. The layer is trained to minimize the cross entropy function [25] given by:

$$E = \frac{1}{N} \sum_{n=1}^{N} \sum_{c \in C} \hat{d}_{cn} \ln d_{cn} + (1 - \hat{d}_{cn}) \ln(1 - d_{cn}) \quad (4)$$

where $\mathbf{d}_n \in \mathbb{R}^{31}$ is the true probability of the n-th sample belonging to a specific class.

Once the process of training the autoencoders and the softmax layer, the network undergoes a last refined training phase. In this last process, only the first, third, and fifth layers are fully connected and trained simulating a feed-forward neural network, as shown in Figure 3. The initial weights are those obtained during the encoding-decoding learning phase and fine tuned using SCG descent to minimize the cross entropy in (4).

4. Experiments

4.1 Setup

The experimental setup was as follows. The dataset was divided in 50%, 25%, and 25% for training, testing, and validation respectively. Two autoencoders were trained using SCG to minimize the cross entropy previously defined. The first autoencoder has 100 neurons and the second one has 50. The second autoencoder is trained with the reduced features from the first autoencoders. A softmax neural network is then trained to minimize the mean squared error with SCG.

The process is done once per each subject to match the experiments done by Kang et. al. in [4]. The data was not processed in any other manner before being presented to the neural architecture for training nor testing. Thus, the inputs are row-vector versions of 8-bit depth images as shown in Figure 2.

4.2 Results and Discussion

Performance results on the validation sets are shown in Table 1. The table indicates that the average accuracy was of 98.9% which is similar to the results reported in [4] of 99.9%. Although the difference is 1%, it is not statistically significant. Furthermore, the deep architecture proposed in this paper is more efficient than the CNN

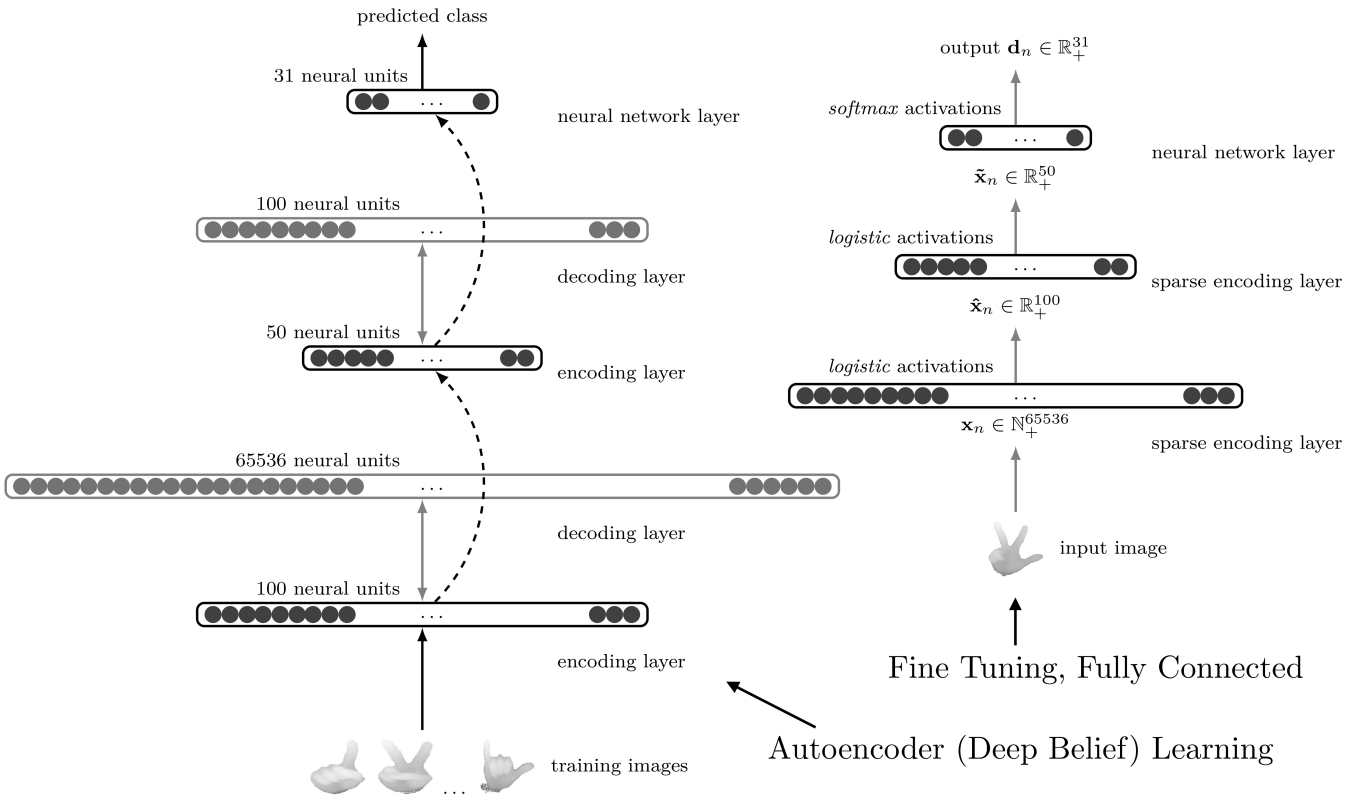

Fig. 3: (*Left*) Deep architecture during training. The training begins with the first two layers, encoder - decoder, and once the training is complete feature are encoded and propagated to the third layer in order to further encode - decode high-level features. Finally, the last neural layer retrieves the encoded features from the third layer and learns the target class. The training is performed using SCG descent [21]. (*Right*) The working architecture when testing the system. The input is any test image which is passed through the first sparse encoder in the stack, reducing the dimensionality of the problem to 100. In the second sparse encoder the feature space is further reduced to 50. The output are probabilistic outputs corresponding to each of the 31 classes.

Table 1: Performance of the deep architecture over the validation sets.

	S1	S2	S3	S4	S5	Avg.
ACC	0.9748	0.9923	0.9935	0.9929	0.9910	0.9889
SPC	0.9991	0.9997	0.9998	0.9998	0.9997	0.9996
MAE	0.1483	0.0640	0.0373	0.0347	0.0494	0.0667

proposed by the authors. The accuracy (ACC) is calculated as $\frac{TP+TN}{TP+FP+TN+FN}$ where TP is the count of true positives, TN is the count of true negatives, FP is the count of false positives, and FN is the count of false negatives. When it comes to specificity (SPC), defined as $\frac{TN}{FP+TN}$, the proposed model gives a measure of 0.99, indicating that the model is very good at predicting true negatives. This quality is desired in this type of problems to make sure the model is sure what a sign is not. This property could be attributed to the minimization of the Kullback-Leibler divergence function defined in (2) that causes the average outputs of the neural units to be low. Similarly, a low mean absolute error (MAE), defined as $\frac{1}{N}\sum_{i=1}^{N}|d_i - \hat{d}_i|$, is an indicator of low average error and low uncertainty in the output of the nerual units. The average MAE was of 0.06 and it was as slow as 0.03 for some subjects.

Figure 4 shows all the errors made after training over the validation set. The figure indicates that the most common error is with the sign for the letter 'E' which is confused with the letters 'S', 'M', and 'T'. A close inspection of Figure 1 reveals that these signs are very similar and these errors are not surprising. Similarly, the sign 'M' was misclassified as 'A' and 'S'. Most misclassifications in Figure 4 are related to signs done when the hand is making a fist form and the differences in depth are low.

To illustrate this further, Figure 5 shows a variety of misclassification errors. From the figure it can be seen that hand gestures with a closed fist can be confused with other similar gestures. However, a close inspection of other gestures suggests that errors are also caused by poor segmentation of the hand area. For example, consider the signs corresponding to numbers '4' and '6', and also letters 'E', 'A', and 'M',

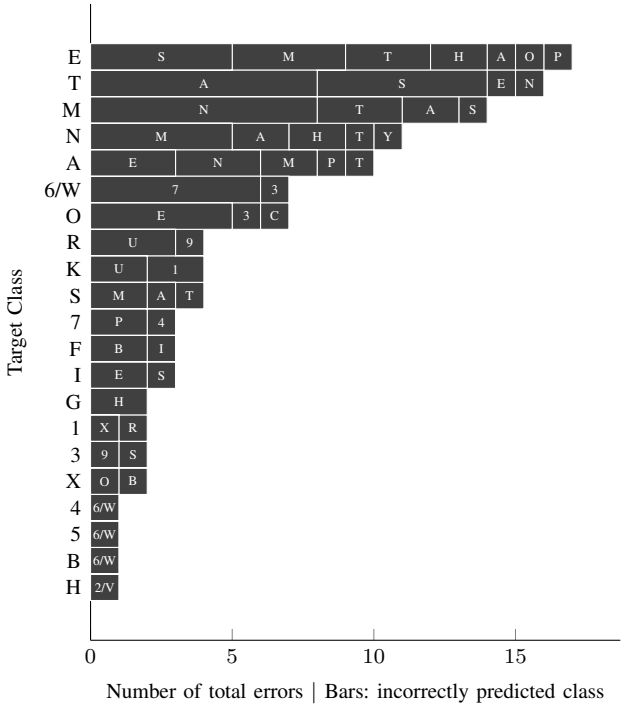

Fig. 4: Stacked bar chart of classification errors. Every stacked bar indicates the predicted class and the size indicates frequency amount. Common errors are with hand signs that involve a closed hand gesture.

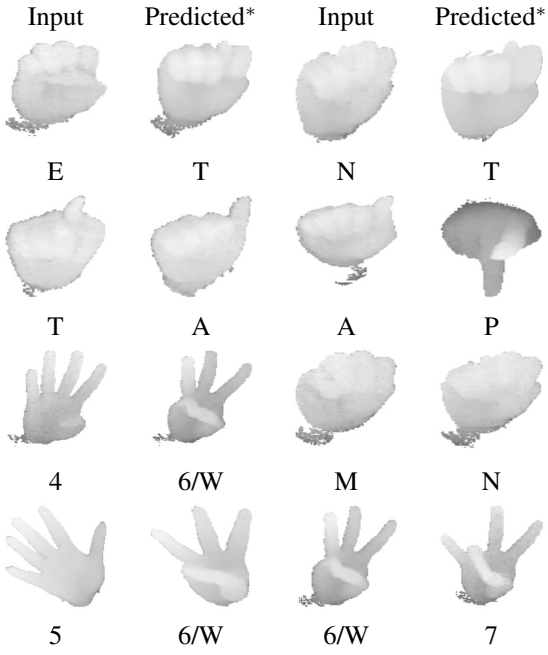

Fig. 5: Examples of classification errors. *The image shown as predicted is the nearest neighbor found in the predicted class with respect to the input.

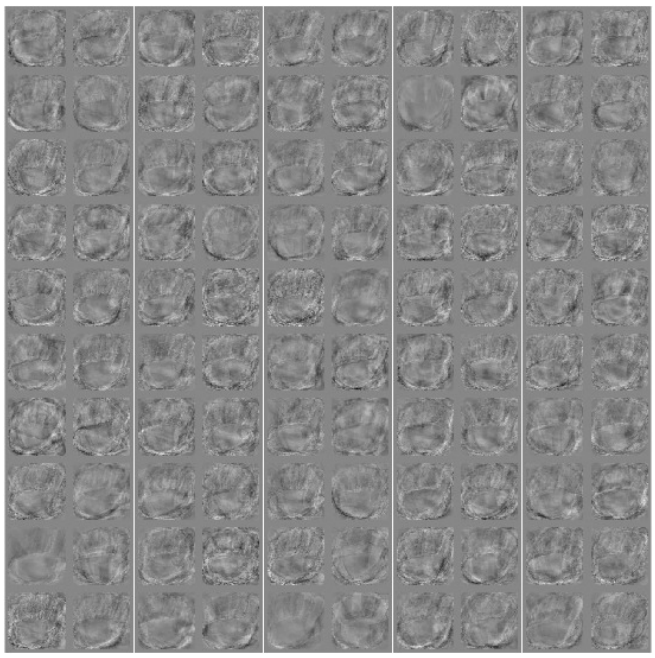

Fig. 6: Weights associated with 100 trained neurons in the first layer for the fifth subject.

the lower portion of the image has a section that was not segmented out of the image and it has been introduced as part of the input. It is possible that the network encoded this as a distinctive feature of the images leading to classification errors. Figure 6 visualizes the weights associated to each neural unit in the first encoding layer for the fifth subject. The weights are capturing higher-level features and at the same time some seem to discard information around the limits of the images, but some seem to rely on regions of the image that are consistent with regions where images exhibit poor segmentation. Further studies are required to confirm this, however.

As shown in Table 1, the overall accuracy is 98.9%, and furthermore, Figure 7 shows the average accuracy for each individual sign. From the figure it can be seen that signs that involve hand gestures are likely to have lower accuracy. The figure also shows that about half of the signs achieve perfect classification over samples that were unseen by the network architecture. This indicates superior generalization abilities of the proposed deep architecture.

Table 2 shows the state of the art on methodologies that take on the general task of classifying hand gestures using different approaches. The reported work is organized by date of publication. The methodologies include work that uses neural networks, *i.e.*, [15], [13], [14], and [4], as well as work that report using other image processing algorithms and non-neural classifiers, *i.e.*, [16], [18], and [19]. Among these important work, some exclusively classify hand gestures, *i.e.*, [13] and [14], others identify only alphabets, *i.e.*, [15],

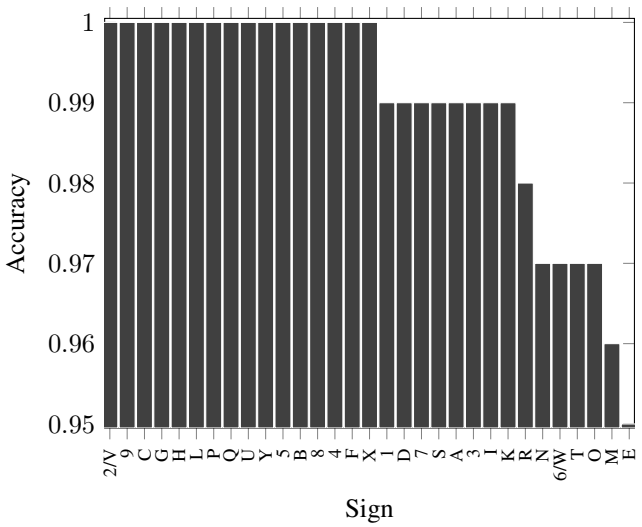

Fig. 7: Average accuracy for each different sign on validation set. Most signs with lower accuracy are ones whose sign involves a closed hand gesture.

Table 2: In some of the reported experiments, one subject was left out of the training set and was used for testing purposes, we refer here to that as leave-one-out (indicated with *).

Ref.	Approach	Year	Cls Typ	$\|C\|$	Inpt Typ	ACC
[15]	FFNN	2004	Alp.	24	Clr Img	0.999
[13]	CNN	2011	Gstrs	6	Clr Img	0.968
[14]	ANMM	2011	Gstrs	6	D. Img	0.991
[16]	Gabor+RF	2011	Alp.	24	D. Img	0.69
[16]	Gabor+RF	2011	Alp.	24	Clr+D.	0.75
[16]	Gabor+RF	2011	Alp.	24	Clr+D.	0.49*
[18]	3D+MLRF	2013	Alp.	24	D. Img	0.87
[18]	3D+MLRF	2013	Alp.	24	D. Img	0.57*
[19]	Jnt Inf+RF	2015	Alp.	24	D. Img	0.90
[19]	Jnt Inf+RF	2015	Alp.	24	D. Img	0.70*
[4]	Deep CNN	2015	Alp.+Dg.	31	D. Img	0.999
[4]	Deep CNN	2015	Alp.+Dg.	31	D. Img	0.855*
ours	Deep AE	2017	Alp.+Dg.	31	D. Img	0.989
ours	Deep AE	2018	Alp.+Dg.	31	D. Img	0.855*

[16], [18], and [19], while [4] and our work focuses on the classification of both alphabets and digits. The work is reported using the standard accuracy measured over cross-validated datasets using all subjects. However, in other cases such as [16], [18], [19], [4], and our work is reported using the leave-one-out (loo) technique on the number of subjects. This indicates that out of all the number of subjects, one was left out for testing while the rest were used for training, and the average accuracy is reported.

Our research indicates that, as shown in Table 2, deep autoencoders have the capability of matching the performance of a convolutional approach. Our main point is that a convolutional approach, while is adequate and performs well, is an *expensive* measure to a problem that may have a simpler deep learning solution, such as an autoencoder. Here when we say expensive, we refer to the amount of computations required to produce a solution using a convolutional neural network. It is known that CNN-based architectures suffer from having a massive amount of parameters to calculate during training [26] and, upon deployment, one needs to sacrifice accuracy for efficiency. Therefore, one of the major points of improvement is to make the number of computations as low as possible while providing good accuracy [25].

The most popular approaches, such as GoogLeNet or ResNet, rely on novel advances to make good convolutional approaches more affordable by dropping neural connections, reducing filter sizes, or experimenting with different architectures [27]. However, some very good implementations of CNNs can cost 358 millions of operations [28] and even 854 millions of operations if implemented naively. Thus, we make the case that, although CNNs are superior to other approaches, one must try other simpler approaches such as deep autoencoders that offer a simple approach to the computational complexity of the deployed models. Our simple approach to the problem only involves 260 million of computations.

Autoencoders not only offer a computationally simpler approach, but also when they are stacked in layers, they are known to learn complex manifold structures and non-trivial patterns existing in the data. Furthermore, they are not biased toward learning to classify a specific group of targets from the data, and thus they belong to a group of neural networks known as deep belief networks [29]. Once the autoencoder is trained without bias, then it is fine-tuned to classify correctly into specific groups, this fine-tuning process was explained in Section 3.3.

For further reference, we performed an experimental search for the best number of layers and the neurons in the autoencoding layers. The search was performed using an exponentially increasing number of neurons. We determined that four layers (two pairs of encoding-decoding layers) was sufficient to perform well in this dataset. Once it was determined that four layers was sufficient, we performed another search more closely looking for best number of neurons in every layer. The best number of neurons was determined to be of 4096 (or 2^{12}) in the first encoding layer, and 2048 (or 2^{11}) in the second, since the error was 0.0027.

Using the number of layers and neurons discussed above, in Table 2 we reported our total accuracy rate to be 0.9889 using all images from all subjects and separating them in 50%, 25%, and 25% for training, validation, and testing, respectively, choosing the samples at random every time, repeating the experiment every time and reporting the average accuracy with a standard deviation of 0.0110. Furthermore, if we repeat the same experiment reported in [4] that consist of training the model with all the images of four subjects and leaving the fifth subject for testing, we report an average accuracy rate of 0.855. This accuracy is averaged across five

runs varying the subject left out every time, leading to a standard deviation of 0.009.

5. Conclusions

We have presented a deep learning architecture based on sparse autoencoders for the problem of sign recognition. The model recognizes 31 different signs in the American Sign Language with an average accuracy of 98.9%. We proposed an architecture with 100 and 50 neural units trained in unsupervised encoding-decoding steps, and a softmax network with 31 units in the output layer corresponding to each class. The input consists of depth-sensor images from five different subjects whose hand sign was segmented to preserve only the regions corresponding to the hand. Further inspection of classification errors indicates that errors are most common among signs that are similar and are among the type that has a closed hand sign; other causes or misclassification come from poor segmentation in the original dataset.

The proposed five layer architecture captures well, in three levels of abstraction, the intricacies of hand gestures. However, the problem of finding a minimal number of neurons without loss of generalization in each layer requires further research. The next steps, however, are to explore subject vs subject experiments, such as training with a varying number of subjects and testing with new unseen subjects whose hand size, shape, and musculoskeletal systems are completely unknown.

Acknowledgements

This work was supported by the New York State (NYS) Cloud Computing and Analytics Center (CCAC), and by the National Council for Research and Technology (Conacyt).

References

[1] T. Starner, J. Weaver, and A. Pentland, "Real-time american sign language recognition using desk and wearable computer based video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1371–1375, 1998.

[2] T. Starner and A. Pentland, "Real-time american sign language recognition from video using hidden markov models," in *Motion-Based Recognition*. Springer, 1997, pp. 227–243.

[3] M. W. Kadous and C. Sammut, "Classification of multivariate time series and structured data using constructive induction," *Machine learning*, vol. 58, no. 2, pp. 179–216, 2005.

[4] B. Kang, S. Tripathi, and T. Q. Nguyen, "Real-time sign language fingerspelling recognition using convolutional neural networks from depth map," in *Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on*. IEEE, 2015, pp. 136–140.

[5] J. F. Lichtenauer, E. A. Hendriks, and M. J. Reinders, "Sign language recognition by combining statistical dtw and independent classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 2040–2046, 2008.

[6] O. Al-Jarrah and A. Halawani, "Recognition of gestures in arabic sign language using neuro-fuzzy systems," *Artificial Intelligence*, vol. 133, no. 1-2, pp. 117–138, 2001.

[7] G. Murthy and R. Jadon, "A review of vision based hand gestures recognition," *International Journal of Information Technology and Knowledge Management*, vol. 2, no. 2, pp. 405–410, 2009.

[8] D. C. Mocanu, E. Mocanu, P. H. Nguyen, M. Gibescu, and A. Liotta, "A topological insight into restricted boltzmann machines," *Machine Learning*, vol. 104, no. 2-3, pp. 243–270, 2016.

[9] M. Donini and F. Aiolli, "Learning deep kernels in the space of dot product polynomials," *Machine Learning*, pp. 1–25, 2016.

[10] M. C. Burl and P. G. Wetzler, "Onboard object recognition for planetary exploration," *Machine learning*, vol. 84, no. 3, pp. 341–367, 2011.

[11] E. Michael, "Deep, deep trouble: Deep learning's impact on image processing, mathematics, and humanity," *SIAM News*, vol. 50, no. 04, 2017.

[12] N. Japkowicz, "Supervised versus unsupervised binary-learning by feedforward neural networks," *Machine Learning*, vol. 42, no. 1, pp. 97–122, 2001.

[13] J. Nagi, F. Ducatelle, G. A. Di Caro, D. Cireşan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. M. Gambardella, "Max-pooling convolutional neural networks for vision-based hand gesture recognition," in *Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 342–347.

[14] M. Van den Bergh and L. Van Gool, "Combining rgb and tof cameras for real-time 3d hand gesture interaction," in *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*. IEEE, 2011, pp. 66–72.

[15] J. Isaacs and S. Foo, "Hand pose estimation for american sign language recognition," in *System Theory, 2004. Proceedings of the Thirty-Sixth Southeastern Symposium on*. IEEE, 2004, pp. 132–136.

[16] N. Pugeault and R. Bowden, "Spelling it out: Real-time asl fingerspelling recognition," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1114–1119.

[17] M. I. Chacon and G. R. Alonso, "Wood defects classification using a som/ffp approach with minimum dimension feature vector," in *International Symposium on Neural Networks*. Springer, 2006, pp. 1105–1110.

[18] A. Kuznetsova, L. Leal-Taixé, and B. Rosenhahn, "Real-time sign language recognition using a consumer depth camera," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2013, pp. 83–90.

[19] C. Dong, M. C. Leu, and Z. Yin, "American sign language alphabet recognition using microsoft kinect," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 44–52.

[20] C. Baker and D. Cokely, "American sign language," *A Teacher's Resource Text on Grammar and Culture. Silver Spring, MD: TJ Publ*, 1980.

[21] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8595–8598.

[22] J. M. Joyce, "Kullback-leibler divergence," in *International Encyclopedia of Statistical Science*. Springer, 2011, pp. 720–722.

[23] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[24] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural networks*, vol. 6, no. 4, pp. 525–533, 1993.

[25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[26] P. Tang, H. Wang, and S. Kwong, "G-ms2f: Googlenet based multi-stage feature fusion of deep cnn for scene recognition," *Neurocomputing*, vol. 225, pp. 188–197, 2017.

[27] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, vol. 4, 2017, p. 12.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, *et al.*, "Going deeper with convolutions." Cvpr, 2015.

[29] Y.-l. Boureau, Y. L. Cun, *et al.*, "Sparse feature learning for deep belief networks," in *Advances in neural information processing systems*, 2008, pp. 1185–1192.