A Convolutional Neural Network to Classify American Sign Language Fingerspelling from Depth and Colour Images

Salem Ameen
University of Salford
S.A.Ameen@edu.salford.ac.uk

Sunil Vadera
University of Salford
S.Vadera@salford.ac.uk

Abstract

Sign language is used by approximately 70 million[1] *people throughout the world, and an automatic tool for interpreting it could make a major impact on communication between those who use it and those who may not understand it.*

However, computer interpretation of sign language is very difficult given the variability in size, shape and position of the fingers or hands in an image.

Hence, this paper explores the applicability of deep learning for interpreting sign language. The paper develops a convolutional neural network aimed at classifying fingerspelling images using both image intensity and depth data.

The developed convolutional network is evaluated by applying it to the problem of finger spelling recognition for American Sign Language. The evaluation shows that the developed convolutional network performs better than previous studies and has precision of 82% and recall of 80%. Analysis of the confusion matrix from the evaluation reveals the underlying difficulties of classifying some particular signs which is discussed in the paper.

Keywords – Deep learning; Convolutional Neural Network (ConvNet); Fingerspelling.

1. Introduction

American Sign Language (ASL) (Vicars, 1997) is an example of one handed sign language and a method of spelling words or letters in the American language. According to Wikipedia, from 250,000 to 500,000 deaf people use this sign language. A person who uses ASL needs to use one hand for spelling the letters, which all have a static picture to show the meaning except two letters *J* and *Z*, which both need a motion to convey meaning.

Several previous studies have attempted to develop systems for interpreting fingerspelling using a combination of image processing methods and learning methods. Most of these studies aim to extract relevant features and then use machine learning methods to induce a classifier.

One of the earliest attempts was by Pugeault and Bowden (2011), who used Gabor filters to extract features which were then used to train multi-class random forests to develop a classifier for 24 letters of ASL.

Rioux-Maldague and Giguere (2014) experiment with different types of feature extraction methods in combination with a deep belief network (DBN) for classification. In their first method, they concatenate both depth and intensity data and use this as input to DBN to classify the images. In a second method, they use 16 Gabor filters with four different scales and orientations to extract features which are used as inputs to a DBN. In a third method, they extract features from the main contours of a hand by using three different types of bar filters (vertical, horizontal and diagonal) and then use these as inputs. In a fourth method, they adjust the depth of images according to the distance between the object and the camera, and then combine them with intensity and use these as input to a DBN.

Moreover, there are many other works related to sign language. Moeslund, et al. (2011) addressed different kinds of sign language such as American Sign Language and British Sign Language, concluding that although sign language recognition is in its infancy, ASL is the subject of most of the research to date.

In this paper we explore a different architecture to the above studies; that is we utilise a ConvNet in which intensity and depth information are used as separate inputs.

The paper is organised as follows: Section 2 begins with some background on deep learning and then presents the architecture proposed in this paper, Section 3 presents the empirical trials and Section 4 presents the conclusions.

[1] http://wfdeaf.org/human-rights/crpd/sign-language

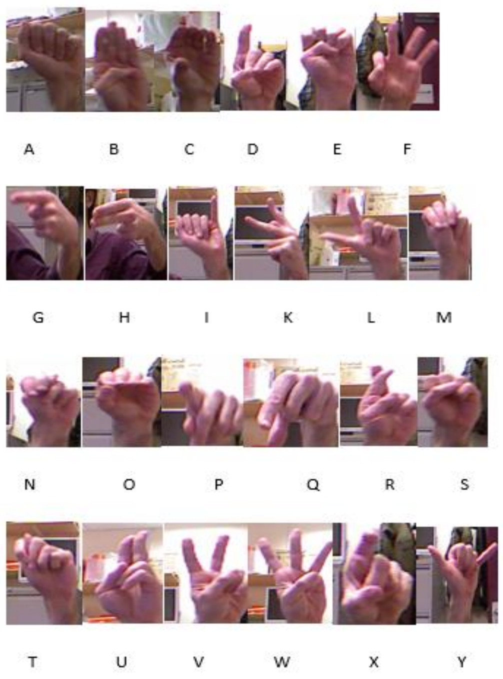


Figure 1. American Sign Language.

2. Architecture for Deep Learning ASL

Although neural networks have a long history (Rumelhart et al., 1986; LeCun et al., 1989), deep learning was introduced fairly recently in the mid-2000s by Hinton and his collaborators (Hinton et al., 2006a; 2006b). As the name suggests, the main idea is to develop a sequence of feature recognition maps, building one layer on top the previous layer and where each layer aims to provide an abstraction of the previous layer, with the final layer performing classification (Yosinski et al., 2014). For example, to recognise objects in images, the first layer aims to learn to recognise edges, the second layer combines edges to form motifs, the third learns to combine motifs into parts, and the final layer learns to recognise objects from the parts identified in the previous layer (LeCun et al., 2015).

This paper aims to utilise such a deep learning architecture to recognise the kind of signs presented as images in Figure 1.

Figure 2 presents a typical architecture for ConvNets which was proposed by (LeCun, Galland et al., 1988, LeCun, Boser et al., 1989, Lecun, Matan, et al., 1990, Lecun, Jackel, et al., 1990, Jackel, Boser et al., 1990, LeCun, Boser, Denker et al., 1990, LeCun, Bengio et al., 1995), where each level contains a convolution module followed by a pooling or subsampling module and a final layer that is a fully connected neural network that performs classification.

In general, a convolution aims to apply kernel transformations on an image to identify relevant features while the main goal of pooling is to introduce invariance to local translation and reduce the number of hidden units (Jarret et al., 2009; Boureau et al., 2010).

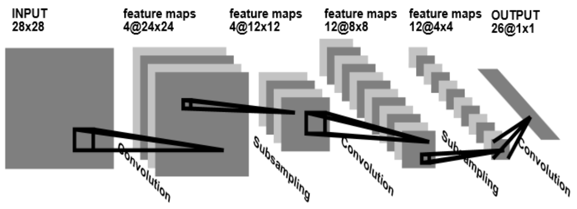


Figure 2. ConvNet (image from LeCun and Bengio, 1995)).

As outlined in the introduction, several authors have tried different feature extraction methods followed by use of learning methods such as random forests. The most promising results to date have been presented by Rioux-Maldague and Giguere (2014) who combine both depth and intensity features and then utilise a DBN for classification.

In this paper, we explore the use of a different architecture that recognises that depth and intensity are inherently different types of information and that there may be advantages in keeping these separate in the initial layers of a ConvNet, leading to the architecture depicted in Figure 3. The following subsections describe the layers of this architecture in more detail.

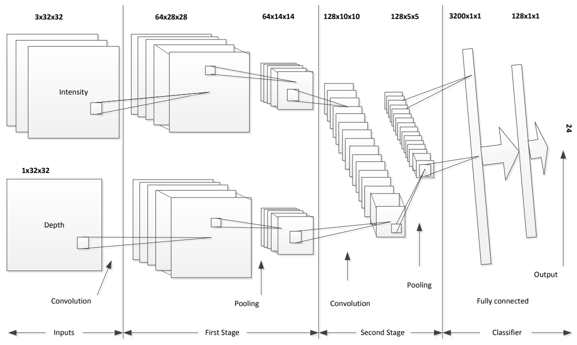


Figure 3. ConvNet model with two inputs (Intensity and Depth).

2.1 Input layer

The input consists of an image of a finger sign in the form of three feature maps (YUV components), each with 32x32 pixels, and one feature map of 32x32 pixels for the depth. Figures 4 shows an example of the normalized input of YUV components of an image and Figure 5 shows an example of the normalised depth data from an image.

Figure 4. Colour images after normalization show three channel in YUV channel.

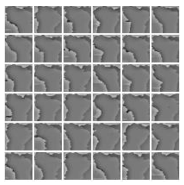


Figure 5. Depth images after normalization.

2.2 Stages 1 & 2: Convolution and pooling layers

Given the normalized representations, convolutions are applied to identify potentially useful features. The convolution of an input **x** with a kernel **k** is computed by (Jarret et al., 2009):

$$(x * k)_{ij} = \sum_{pq=0}^{r-1} (x_{i+p,j+q} k_{r-p,r-q}) \quad (1)$$

Where **x** is the image in the input layer and a feature map in the subsequent layers. The convolution kernel, **k** is a square matrix, and the symbol * denotes the convolution operator. The number of filters in each layer is a hyper parameter that is determined experimentally. In our architecture, 64 filters (feature maps) are used, each with a 5x5 receptive field, no zero padding and a stride of one which leads to 64 planes each of dimension 28x28. In the second stage, 128 filters with the same receptive field and stride are used, leading to an array of 128x10x10. Each single number in this dimension is squashed using a Tanh as an activation function.

In the first stage, a pooling operation is applied to reduce the impact of translations and reduce the number of weights that would be needed.

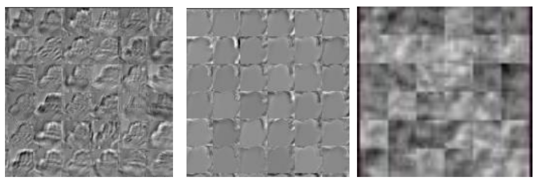
A range of pooling operations have been used in the literature including averaging (Jarrett et al., 2009), maximum value (Boureau, et al., 2010) and Lp-pooling (Sermanet et al., 2012). Following some preliminary experimentation with these, the Lp-Pooling operation, which is defined by the following was adopted:

$$O = \left(\sum_i \sum_j I(i,j)^P \times G(i,j)\right)^{\frac{1}{P}} \quad (2)$$

Where I and O are the input and output respectively and G is a Gaussian kernel. P is hyper parameter that needs to be tested on validation data.

Following pooling, a convolution is again applied to the intensity and depth arrays.
In this experiment, the 64 filters are pooled by a 2x2 receptive field with a stride of 2, leading to 64 planes each of dimension 14x14. In the second stage, the 128 filters are pooled by a 2x2 receptive field with stride of 2, leading to 128x5x5 planes.

The output from the first layer provides a representation of the edges of the depth and intensity. Figure 5(a) and 5(b) show examples of the output from the first layer and Figure 5(c) shows a typical output from the second layer. As these images show, the first layer mainy produces edges while the second layer combines the edges to start forming objects.



(a) (b) (c)

Figure 6. The output of convolutional layer: (a) and (b) after the first convolutional layer of RGB and depth respectively. (c) after the convolution in the second stage.

2.3 Stage 3: Classification Layer

The final layer aims to perform the classification using a fully connected feedforward neural network.

The 128-dimensional feature vectors with a matrix of size 5x5 is reshaped to a single 3200 dimensional vector, and used as input to a two-layer neural net with 128 nodes in the hidden layer and 24 class nodes, one for each letter.

3. Empirical Evaluation

As mentioned earlier, some of the best results for recognising ASL have been obtained by adopting the architecture presented in (Rioux-Maldague and Giguere, 2014) and the aim of this paper has been to try the alternative architecture presented in Figure 3. The architecture was implemented using the Torch scientific computing framework (Collobert et al., 2012). To enable comparison, the same experimental methodology as (Rioux-

Maldague and Giguere, 2014) is adopted. That is, given n users, a model is first developed using the data from the first n-1 users and tested on the nth user. Next, a model is trained on all the data except the (n-1)th and tested on the (n-1)th user, etc. This results in n values which are averaged to produce estimates of the precision and recall measures.

To enable comparison, the same ASL fingerspelling data is used as in (Pugeault and Bowden, 2011; Rioux-Maldague and Giguere, 2014). The dataset represents images of the fingerspelling alphabet of ASL by five different users A, B, C, D and E. The dataset contains all the letters except letters J and Z as both of these need motion. The dataset contains more than 60000 images and there are more than 500 images for each particular sign for each user.
The results were evaluated by computing the recall and precision measures for each letter and comparing the results to the best in class for this benchmark, which is the study by Rioux-Maldague and Giguere (2014). The experiments were run for 250 epochs or until the neural network converged.

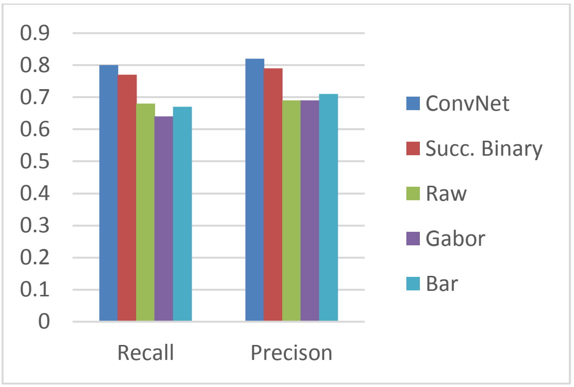


Figure 7. Comparison of the recall and precision for the different types of features used in (RiouxMaldague and Giguere, 2014) and ConvNet

Appendices B and C present the precision and recall for all classes separately and Appendix D shows the confusion matrix of the results. In addition, Appendix A shows the classification accuracy and f1 score of the model.

Figure 7 compares the results of the ConvNet architecture used in this paper with the feature extraction methods and use of DBN presented in (Rioux-Maldague and Giguere, 2014), namely:

(i) **Succ. Binary** method, that adjusts the depth to be correlated with intensity in the input level.

(ii) **Raw** which is based on using a combination of the raw intensity and depth data.

(iii) **Gabor** which uses 16 Gabor filters to extract features.

(iv) **Bar** which is based on using three bar filters to extract features of the main contours from an image of a hand.

Table 1 compares the results from the Convnet architecture developed in this paper with the best results to date, which are presented in Rioux-Maldague and Giguere(2014) .

	Precision	**Recall**
Rioux-Maldague and Philippe Giguere	79%	77%
ConvNet	82%	80%

Table 1. Precision and Recall results

Why might the results from the use of ConvNet be better? One possible explanation is that in the architecture presented in this paper, the first stage has two separate parts: one extracts the edges of RGB images, the other extracts the edges of the depth. The features are then combined in the second stage. In contrast, the existing approach combines the depth and intensity information in the first phase, resulting in a much bigger search space of potential features of which only a subset will be meaningful.

However, the performance of the architecture used in this paper is not uniform across all the letters as shown in Figures 8 and 9 which compares the results of recall and precision with the Succ. Binary method.

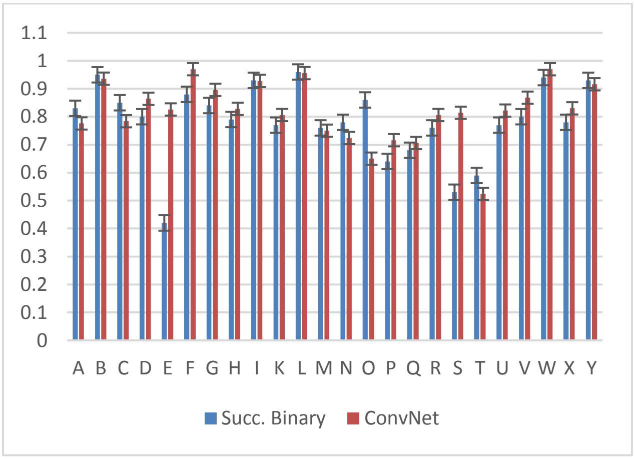


Figure 8. Precision with Standard Error

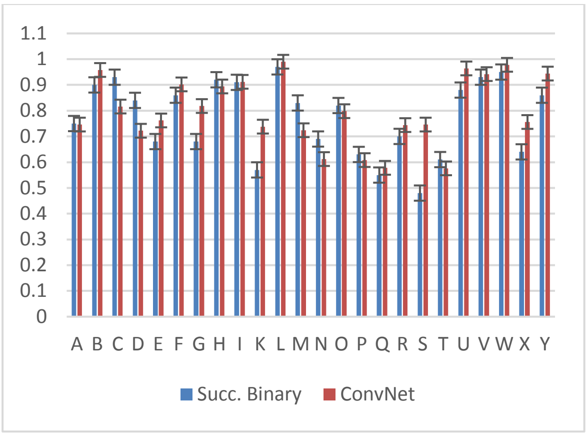
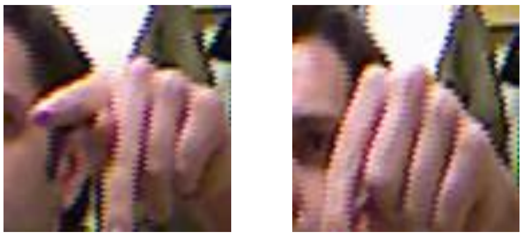


Figure 9. Recall with Standard Error

The performance of the model for the letters F, W, L, B, I and Y is very good with more than 90% recall and precision. The model's performance on the letter T is only 52% for recall and 58% for precision. Nevertheless, the results show that the model is more robust to confusion between letters when compared to previous work, where letters like E, P and K have less than 50% for recall. Examining the confusion matrix in Appendix D, we can identify cases where the classification is weakest. These include the cases shown in Figure 10 such as P&Q, T&A and G&H, where there is mutual confusion in classification. There are also asymmetric cases, such as those shown in Figure 11, where for example, Q is misclassified as O, and R is misclassified as U, though rarely the other way round. The letter Q has the most variation in the dataset. Figure 12 shows examples of how different users represent the shape of the letter Q and compares them with the recommended shape (image on the left). Another interesting observation is that the sign for the letter R has nearly the same shape as that for the letter U, especially when the hand moves. In both letters, the signer needs to use two fingers to convey the meaning. In addition, the distance between the camera and the fingers is nearly equal which makes it difficult to recognize the differences even when using depth.



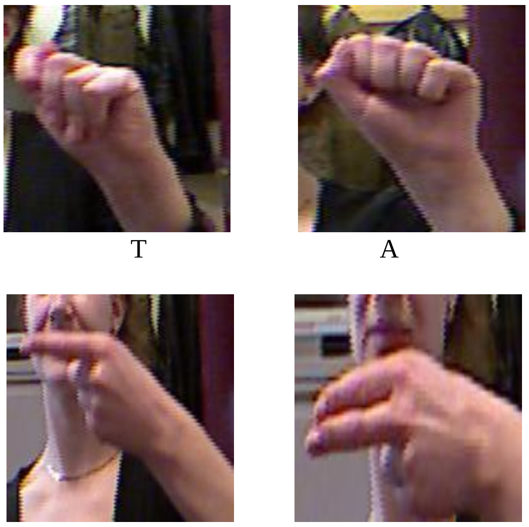


Figure 10. Symmetric confusion

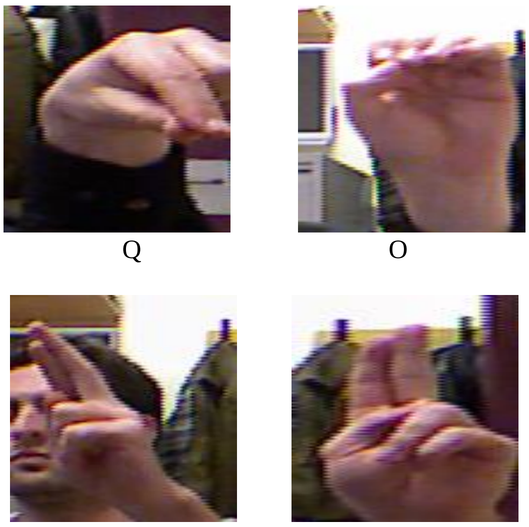


Figure 11. Asymmetric confusion

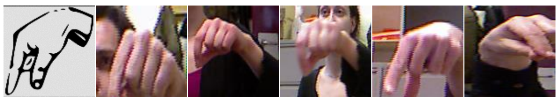


Figure 12. The sign variations of letter Q. Image on the left depicts the recommended shape for the letter Q [from [2]]

4. Conclusion and Future Work

The ability to automatically recognise sign language could have a major impact on the lives of

[2]http://www.tuxpaint.org/stamps/index.php3?cat=symbols&perpage=25

many people who use it to communicate. However, developing systems that recognise signs from images is a challenging task given the variation in size, position, and shapes adopted by different people. Several authors have studied the development of systems that aim to automatically recognise sign language by using feature extraction methods followed by machine learning methods to learn classification models with varying degrees of success.

This paper has developed an alternative architecture that takes both depth and intensity information as different types of inputs to develop a ConvNet. The ConvNet was implemented in Torch and evaluated on a benchmark American Sign Language data set. The results of the empirical evaluation show an improvement of about 3% compared to previous work, with recall and precision rates over 80%. An analysis of the confusion matrix has identified two type of errors: (i) symmetric errors, such as two letters that can be misclassified as each other and (ii) asymmetric errors, where one letter is misclassified as another but not the other way round. Future work to improve accuracy could include using transfer learning (Yosinski et al., 2014; Razavian et al., 2014) where a pre-trained model like AlexNet (Krizhevsky et al., 2012), VGGNet (Simonyan and Zisserman, 2014) or any pre-trained model from Caffe zoo[3] can provide a good initial model, and use data augmentation methods (He et al., 2015) to increase the volume of data available for training. To enable comparison with related studies, segmentation and background removal methods were not adopted, which if applied, can be expected to result in further improvements to the overall accuracy.

Reference:

BOUREAU, Y.-L. , PONCE, J. and LECUN, Y. (2010) A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning,* pp. 111-118.

COLLOBERT, R., KAVUKCUOGLU, K. and FARABET, C. (2012) Implementing neural networks efficiently. In *Neural Networks: Tricks of the Trade*, ed: Springer, pp. 537-557.

HE, K., ZHANG, X., REN, S. and SUN, J. (2015) Deep Residual Learning for Image Recognition, *arXiv preprint arXiv:1512.03385*.

HINTON, G., OSINDERO, S. and TEH, Y. (2006a) A Fast Learning Algorithm for Deep Belief Nets, *Neural Computation,* vol. 18, pp. 1527-1554.

HINTON, G. E. and SALAKHUTDINOV, R.R. (2006b) Reducing the dimensionality of data with neural networks, *Science,* vol. 313, pp. 504-507.

JACKEL, L., BOSER, B., DENKER, J. , GRAF, H., LECUN, Y., GUYON I.*, et al.*, (1990) Hardware requirements for neural-net optical character recognition. In*, International Joint Conference on Neural Networks*, 855-861.

JARRETT,K., KAVUKCUOGLU, K. , RANZATO, M. and LECUN, Y. (2009) What is the best multi-stage architecture for object recognition? In *IEEE 12th International Conference on Computer Vision,* 2146-2153.

KRIZHEVSKY, A. , SUTSKEVER, I. and HINTON, G. E. (2012) Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097-1105.

LECUN, Y. , BOTTOU, L. , ORR, G. and MÜLLER, K.-R. (1989) Efficient BackProp. In *Neural Networks: Tricks of the Trade*. vol. 1524, G. Orr and K.-R. Müller, Eds., ed: Springer Berlin Heidelberg, 9-50.

LECUN, Y. , BENGIO, Y. and HINTON, G. "Deep learning, (2015) *Nature,* vol. 521, 436-444.

LECUN, Y., GALLAND, C. C. and HINTON, G. E. (1988) GEMINI: Gradient Estimation Through Matrix Inversion After Noise Injection. In *NIPS*, pp. 141-148.

LECUN, Y., BOSER, B. , DENKER, G. E., HENDERSON, D. , HOWARD, R. E. ,. HUBBARD*,* W. *et al.* (1989) Backpropagation applied to handwritten zip code recognition, *Neural computation,* vol. 1, pp. 541-551.

LECUN, Y. , JACKEL, L. , BOSER, B. , DENKER, J. , GRAF, H. , GUYON*,* I. *et al.* (1990) Handwritten Digit Recognition: Applications of Neural Net Chips and

[3] https://github.com/BVLC/caffe/wiki/Model-Zoo

Automatic Learning, *Neurocomputing*, Springer, 303-318.

LECUN,Y., BOSER, B., DENKER, J. S. , HOWARD, R. E. , HUBBARD, W., L. D. JACKEL, and HENDERSON, D. (1990) Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, 396-404.

LECUN, Y., MATAN, O., BOSER, B. , DENKER, J., HENDERSON, D. , HOWARD*,* R. *et al.* (1990) Handwritten zip code *recognition* with multilayer networks. In *Proceedings. 10th International Conference on Pattern Recognition*, 35-40.

LECUN, Y. and BENGIO, Y. (1995) Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks,* vol. 3361.

MOESLUND, T. B., HILTON, A., KRÜGER, A. and SIGAL,L. (2011) *Visual Analysis of Humans*: Springer.

PUGEAULT, N. and BOWDEN, R. (2011) Spelling it out: Real-time ASL *fingerspelling* recognition. In *IEEE International Conference Workshop on Computer Vision,* 1114-1119.

RAZAVIAN, A. S. , AZIZPOUR, H., SULLIVAN, J. and CARLSSON, S. (2014) CNN Features off-the-shelf: an Astounding Baseline for Recognition. In IEEE *Workshop on Computer Vision and Pattern Recognition*, 512-519.

RIOUX-MALDAGUE, L. and GIGUERE, P. (2014) Sign Language Fingerspelling Classification from Depth and Color Images Using a Deep Belief Network. In *Canadian Conference on Computer and Robot Vision (CRV),* 92-97.

RUMELHART, D. E., HINTON, G. E. and WILLIAMS, R. J. (1986) Learning representations by back-propagating errors, *Nature,* vol. 323, pp. 533-536.

SERMANET, P. , CHINTALA, S. and LECUN, Y. (2012) Convolutional neural networks applied to house numbers digit classification. In 21st International Conference on Pattern Recognition 3288-3291.

SIMONYAN, K. and ZISSERMAN,A. (2014) Very deep convolutional networks for large-scale
image recognition, *arXiv preprint arXiv:1409.1556.*

VICARS, W. American Sign Language, 1997

YOSINSKI, J. , CLUNE, J., BENGIO, Y. (2014) and LIPSON, H. "How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, 3320-3328.

Appendix A

Table 2 shows classification accuracy and f1 score on our model. The first column shows training of users A, B, C, D and testing on user E and the second column shows training one users A, B, C, E and testing on D and so on.

	Testing on user:					Average
	E	D	C	B	A	
Accuracy	83.65%	71.29%	87.70%	80.01%	79.06%	80.34%
F1 score	82%	70%	87%	79%	78%	79.20%

Table 2. The accuracy and f1 score of the model

Appendix B

Table 3 shows the precision in all classes and the average precision over the models testing on user E, D, C, B and A.

	Precision					
	Test on user:					Average per Class
class	E	D	C	B	A	
A	74%	93%	95%	44%	82%	**78%**
B	96%	90%	97%	91%	94%	**94%**
C	87%	56%	92%	90%	67%	**78%**
D	96%	71%	79%	92%	94%	**86%**
E	67%	91%	96%	94%	65%	**83%**
F	98%	97%	99%	93%	98%	**97%**
G	70%	99%	97%	90%	92%	**90%**
H	97%	74%	100%	71%	72%	**83%**
I	100%	98%	92%	84%	90%	**93%**
K	96%	93%	97%	68%	49%	**81%**
L	100%	95%	92%	92%	99%	**96%**
M	81%	59%	90%	58%	87%	**75%**
N	90%	20%	76%	95%	81%	**72%**
O	51%	41%	79%	84%	70%	**65%**
P	88%	46%	68%	86%	70%	**72%**
Q	83%	55%	85%	76%	54%	**71%**
R	93%	89%	66%	76%	79%	**81%**
S	80%	77%	90%	66%	94%	**81%**
T	52%	43%	71%	43%	53%	**52%**
U	97%	54%	96%	95%	69%	**82%**
V	89%	75%	97%	81%	92%	**87%**
W	98%	94%	98%	96%	99%	**97%**
X	91%	85%	81%	87%	71%	**83%**
Y	83%	80%	99%	97%	99%	**92%**
Average Precision	**86%**	**74%**	**89%**	**81%**	**80%**	***82%***

Table 3. Precision. The cell with red colour means that the precision is under 20% and with yellow colour means the precision is between 20% and 40%.

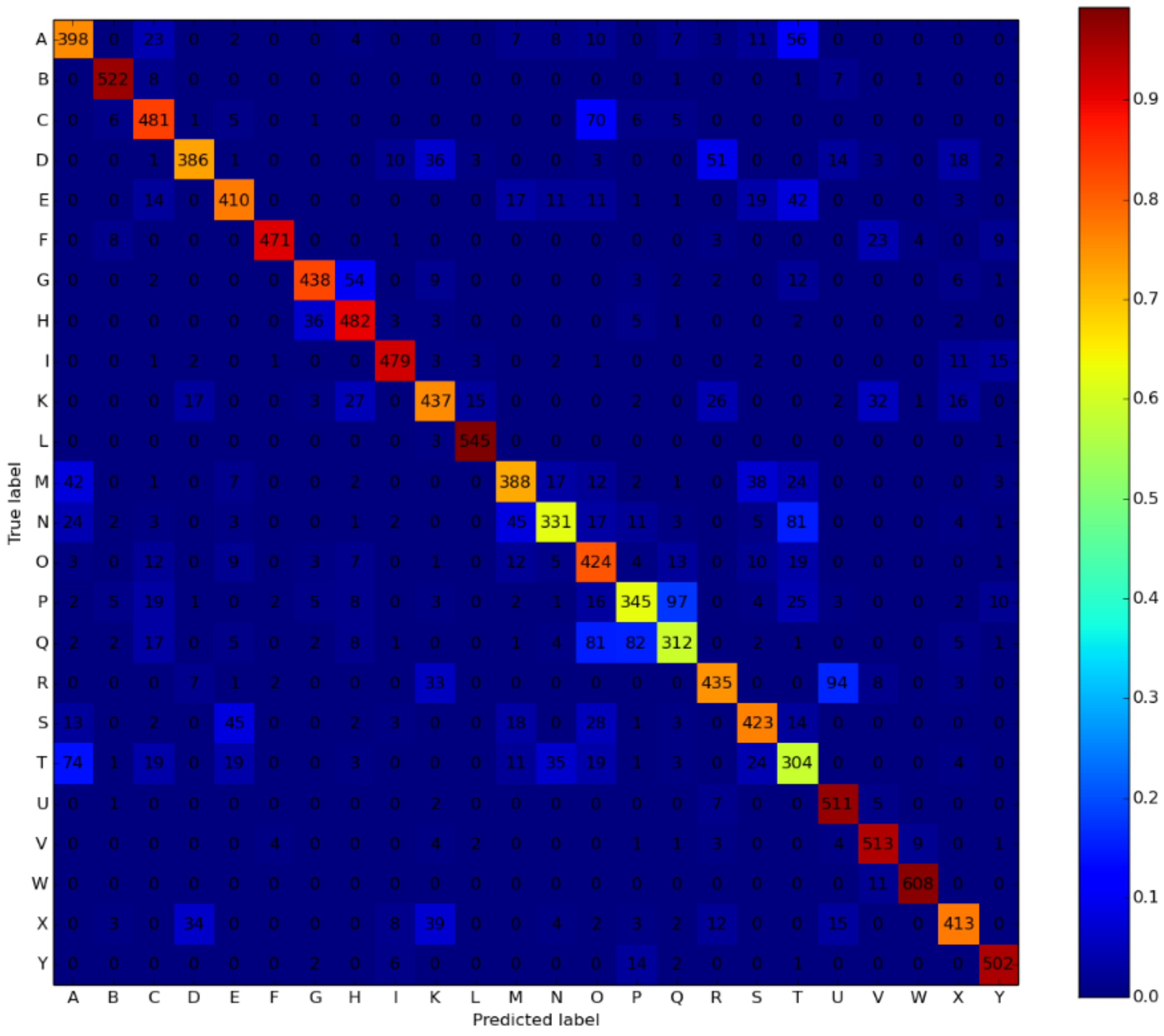
Appendix C

Table 4 shows the recall on all classes and the average recall over the models testing on user E, D, C, B and A.

	Recall					
	Test on user:					Average per Class
class	1	2	3	4	5	
A	71%	67%	55%	80%	100%	**75%**
B	100%	93%	98%	89%	99%	**96%**
C	96%	46%	89%	87%	90%	**82%**
D	86%	82%	68%	49%	76%	**72%**
E	70%	34%	90%	95%	92%	**76%**
F	100%	66%	96%	91%	98%	**90%**
G	83%	62%	99%	67%	98%	**82%**
H	64%	99%	96%	88%	100%	**89%**
I	91%	90%	94%	89%	92%	**91%**
K	88%	94%	41%	79%	67%	**74%**
L	100%	100%	100%	97%	98%	**99%**
M	87%	58%	83%	35%	99%	**72%**
N	76%	8%	94%	77%	51%	**61%**
O	98%	66%	91%	70%	74%	**80%**
P	38%	34%	92%	76%	64%	**61%**
Q	5%	79%	85%	85%	35%	**58%**
R	96%	65%	99%	88%	24%	**74%**
S	98%	58%	89%	85%	43%	**75%**
T	67%	82%	74%	21%	44%	**58%**
U	100%	97%	98%	89%	98%	**96%**
V	100%	95%	92%	85%	99%	**94%**
W	99%	92%	99%	99%	100%	**98%**
X	95%	39%	90%	97%	57%	**76%**
Y	100%	100%	82%	92%	98%	**94%**
Average Recall	**84%**	**71%**	**87%**	**80%**	**79%**	***80%***

Table 4. Recall. The cell with red colour means that the recall is under 20% and with yellow colour means the recall is between 20% and 40%.

Appendix D



	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
A	398	0	23	0	2	0	0	4	0	0	0	7	8	10	0	7	3	11	56	0	0	0	0	0
B	0	522	8	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	7	0	1	0	0
C	0	6	481	1	5	0	1	0	0	0	0	0	0	70	6	5	0	0	0	0	0	0	0	0
D	0	0	1	386	1	0	0	0	10	36	3	0	0	3	0	0	51	0	0	14	3	0	18	2
E	0	0	14	0	410	0	0	0	0	0	0	17	11	11	1	1	0	19	42	0	0	0	3	0
F	0	8	0	0	0	471	0	0	1	0	0	0	0	0	0	0	3	0	0	0	23	4	0	9
G	0	0	2	0	0	0	438	54	0	9	0	0	0	0	3	2	2	0	12	0	0	0	6	1
H	0	0	0	0	0	0	36	482	3	3	0	0	0	0	5	1	0	0	2	0	0	0	2	0
I	0	0	1	2	0	1	0	0	479	3	3	0	2	1	0	0	0	2	0	0	0	0	11	15
K	0	0	0	17	0	0	3	27	0	437	15	0	0	0	2	0	26	0	0	2	32	1	16	0
L	0	0	0	0	0	0	0	0	0	3	545	0	0	0	0	0	0	0	0	0	0	0	0	1
M	42	0	1	0	7	0	0	2	0	0	0	388	17	12	2	1	0	38	24	0	0	0	0	3
N	24	2	3	0	3	0	0	1	2	0	0	45	331	17	11	3	0	5	81	0	0	0	4	1
O	3	0	12	0	9	0	3	7	0	1	0	12	5	424	4	13	0	10	19	0	0	0	0	1
P	2	5	19	1	0	2	5	8	0	3	0	2	1	16	345	97	0	4	25	3	0	0	2	10
Q	2	2	17	0	5	0	2	8	1	0	0	1	4	81	82	312	0	2	1	0	0	0	5	1
R	0	0	0	7	1	2	0	0	0	33	0	0	0	0	0	0	435	0	0	94	8	0	3	0
S	13	0	2	0	45	0	0	2	3	0	0	18	0	28	1	3	0	423	14	0	0	0	0	0
T	74	1	19	0	19	0	0	3	0	0	0	11	35	19	1	3	0	24	304	0	0	0	4	0
U	0	1	0	0	0	0	0	0	0	2	0	0	0	0	0	0	7	0	0	511	5	0	0	0
V	0	0	0	0	0	4	0	0	0	4	2	0	0	0	1	1	3	0	0	4	513	9	0	1
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	608	0	0
X	0	3	0	34	0	0	0	0	8	39	0	0	4	2	3	2	12	0	0	15	0	0	413	0
Y	0	0	0	0	0	0	2	0	6	0	0	0	0	0	14	2	0	0	1	0	0	0	0	502

Figure 13. Confusion matrix of the model