A Study of Convolutional Architectures for Handshape Recognition applied to Sign Language

Facundo Quiroga[1], Ramiro Antonio[1], Franco Ronchetti[1], Laura Lanzarini[1], and Alejandro Rosete[2]

[1] Instituto de Investigación en Informática LIDI, Facultad de Informática, Universidad Nacional de La Plata
{fquiroga,fronchetti,laural}@lidi.info.unlp.edu.ar,
{ramiro.antonio}@outlook.com
[2] Universidad Tecnológica de La Habana José Antonio Echeverría
{rosete}@ceis.cujae.edu.cu

Abstract. Convolutional Neural Networks have been providing a performance boost in many areas in the last few years, but their performance for Handshape Recognition in the context of Sign Language Recognition has not been thoroughly studied. We evaluated several convolutional architectures in order to determine their applicability for this problem.
Using the LSA16 and RWTH-PHOENIX-Weather handshape datasets, we performed experiments with the LeNet, VGG16, ResNet-34 and All Convolutional architectures, as well as Inception with normal training and via transfer learning, and compared them to the state of the art in these datasets. We included experiments with a feedforward neural network as a baseline. We also explored various preprocessing schemes to analyze their impact on the recognition.
We determined that while all models perform reasonably well on both datasets (with performance similar to hand-engineered methods), VGG16 produced the best results, closely followed by the traditional LeNet architecture. Also, pre-segmenting the hands from the background provided a big boost to accuracy.

Keywords: convolutional neural networks, sign language recognition, handshape recognition.

1 Introduction

Automatic sign language recognition (SLR) is an important topic within the areas of human-computer interaction and machine learning that also requires the intervention of various knowledge areas, such as image and video processing, computer vision and linguistics. Robust recognition of sign language could assist in the translation process and the integration of hearing-impaired people, as well as the teaching of sign language for the hearing population.

The problem it tackles is how to convert a video where a signer is speaking in some variant of Sign Language to a text that conveys the same meaning as the signs in the video. SLR implies a pipeline of tasks which starts with the recognition of the shape of the hands of the signer, their movement and position, the shape of lips and face expressions, plus the objects in the scene, in every frame of the video. Afterwards, the signs in the video must be identified, using both information about how signs are made and the syntax and semantics of the sign language. Afterwards, the sentences in sign language must be translated to some target written/spoken language, such as English or Spanish [13, 18]. Von Agris and Cooper present general reviews of the standard techniques of the field [2, 18]

Previous work has shown that the quality of the handshape recognition appears to be the most important factor when recognizing signs [11, 12]. Also, most experiments focused on finding adequate descriptors for the handshapes to be classified with conventional techniques such as SVM or Feedforward Neural Networks. Perhaps the most widely explored descriptors are the geometric descriptors computed on the contour of the hand [18], such as the perimeter, the orientation of the main diagonals, the area, the eccentricity, among others. Another widely used feature for handshape classification is the Histogram of Oriented Gradients (HOG) [3], where the image is divided in a square overlapped grid of small sub-images and it computes a histogram of directions of different gradient in each sub-image. In [13] the Radon transformation is used as a feature for the classification of handshapes.

There have been great advances in the last few years in text, sound, image and video processing by using Convolutional Neural Networks (CNNs), which are simply deep networks that employ convolutional layers [9]. These layers implement a convolution operation and have a sparse connection topology that allows a better processing of images and other signals.

These networks have been recently used for sign language and action recognition [1, 8]. In particular, [8] employs a semi-supervised training procedure with a pretrained Inception [16] network as base classifier to perform handshape recognition, with state-of-the-art results. However, to the best of our knowledge, there has been no systematic study of the applicability of CNNs for handshape recognition. This paper presents experiments on the applicability of various CNNs models (LeNet, Inception, All Convolutional, VGG, ResNet) to the problem of handshape recognition for sign language, using datasets LSA16 [13] and RWTH-PHOENIX-Weather [5].

2 Architectures

In this section we will introduce the convolutional architectures used to compare their performance on the task of handshape recognition. We have chosen a simple convolutional models like LeNet, one of the first convolutional networks, as baseline and also more complex and deep models that have proved their performance on image recognition problems in known datasets such as All Convolutional, VGG, Inception-v3 and ResNet.

2.1 LeNet

The LeNet model was proposed by LeCun et al.[10] in 1998. On their paper, the authors propose a convolutional architecture for OCR and character recognition in documents. It is one of the first convolutional neural networks and a common baseline in the field of Deep Learning.

A standard 2D convolutional layer applies a convolutional operation on an input feature map with two spatial dimensions and a filter/channel dimension. The convolutional mask has size Height*Width*Channels, and is learnable. A layer typically has many such convolutional masks, which determine the number of output channels/filters. Convolutions can be strided to perform downsampling in the same operation.

A max-pooling layer reduces the spatial size of the input to reduce the amount of parameters and computations in the network.

This architecture consists of two sets of convolutional, activation and max-pooling layers followed by two fully-connected layers and finishing with a softmax classifier.

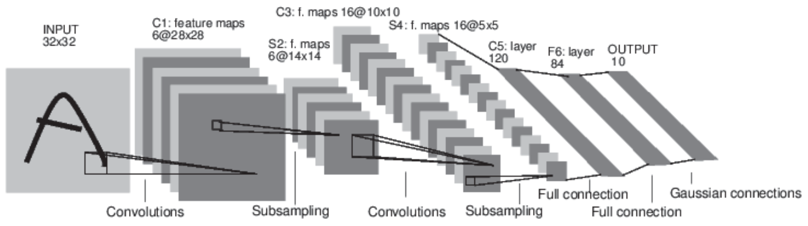

Fig. 1. Diagram of the LeNet5 architecture.

2.2 All Convolutional

This architecture was introduced by Springenberg et al.[15] in 2014. Their model is based on removing the pooling layers of a traditional convolutional network such as LeNet. The spatial downsampling is performed instead by using convolutional layers with a stride greater than 1. The model can achieve state of the art performance on several object recognition datasets such as CIFAR-10, CIFAR-100 and ImageNet, which provides some evidence for the hypothesis that the max pooling operation's contribution to a network comes mostly from the downsampling effect, and not from the maximum computation.

2.3 VGG

The VGG network was first described by Simonyan and Zisserman in their 2014 paper [14]. Its main contribution is providing evidence in favour of stacking convolutional layers with small filters (3x3) instead of using a single layer with bigger filter sizes (5x5, 7x7), since stacked 3x3 filters can approximate bigger ones. The network therefore consists of a many 3x3 convolutional filters stacked in groups of two or three, with max-pooling layers between them and finishing with two 4096-dimensional fully-connected layers

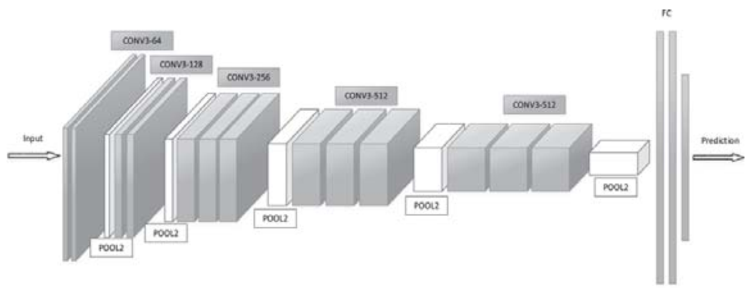

Fig. 2. Diagram of the VGG architecture with its stacked filters with max-pooling layers between them.

This gives a massive number of parameters and learning power. That is why this architecture is costly at runtime and many models that improve it try to reduce the dimensionality of the input to accelerate the computations. The training of VGG is very challenging, specially for the 16 and 19 depth versions. Symonyan and Zisserman trained versions of VGG with less weight layers first. When this smaller networks converged, they were used as initializations for the larger and deeper networks. Two disadvantages of this network are that its training is slow and that its architecture is quite large in terms of memory compared with other state of the art networks.

2.4 Inception

The Inception architecture was introduced by Szegedy et al.[16] in 2014.This original network was called GoogLeNet but later its newer versions were called Inception-vN, with N representing the version named by Google.

The main contribution of this model is the bottleneck layer applied before the parallel convolutions that reduces the computational cost of all the architecture. This bottleneck layer consists employing 1x1 convolutional filters to reduces the spatial input dimension giving a lower number of computations in the subsequent layers.

The goal of this model is to extract features at multiple levels in the same instance of the model. In that way, when a layer of this model receives its input, first, it reduces the input size in a bottleneck layer, and then computes 1x1, 3x3, and 5x5 convolutional filters in parallel. After that, the output of this filters are concatenated in a last filter maintaining the channel dimension. The main goal of this architecture was to reduce the computational cost of the network.

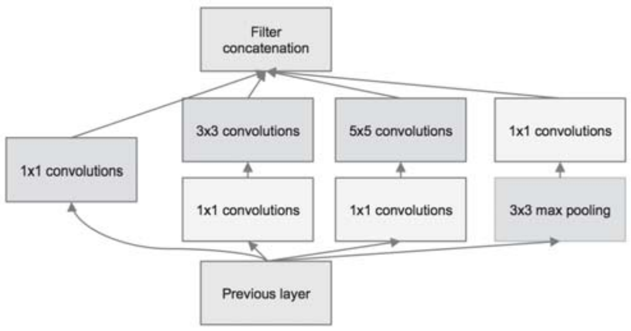

Fig. 3. Diagram of the inception module with its bottleneck layer applied before the 3x3 and 5x5 convolutional filters.

The Inception-v3 was described in 2015 by Szegedy et al.[17] on a paper which proposes updates to the architecture to boost its ImageNet classification accuracy. Even with these updates, the weights of this model are still smaller than those of the VGG model.

2.5 ResNets

This model was presented by He et al.[6] in the year 2015. It presented a modification of the convolutional architecture that allowed training of very deep networks. The problem it addresses is that while adding layers to a network should never lower its performance, since in the worst case the layer should just learn the identity function, in practice networks with many layers. Its main insight was the design of convolutional blocks called *residual modules*, which outputs the sum of the features obtained from the convolutional layers with the input. This design causes residual modules to be initialized to a slightly distorted identity function. Also, the fact that the input is bypassed to the output allows a shortcut path for the gradient to go backwards.

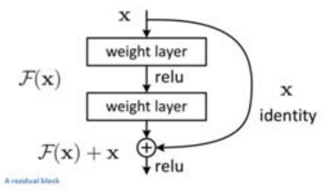
x
weight layer
relu
$\mathcal{F}(\mathbf{x})$
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu
A residual block

Fig. 4. Diagram of the ResNet architecture.

Hence, the most important contribution of this architecture is the evidence of a very deep network having hundreds of layers can actually be trained using standard stochastic gradient descent through the use of residual modules.

The network has 1x1 convolutional filters in the residual modules to form bottleneck layers as seen in the Inception architecture; this helps reducing the dimensionality inside the residual module, and then restoring it, since a hard constraint of this models is that the dimensionality of the output has to match that of the input. ResNet uses a global pooling layer plus softmax as final classifier.

3 Experiments and Results

We performed experiments with the models described in section 2 and two datasets that contain images of the hands of several people with different handshapes.

3.1 Datasets

LSA16 The LSA16 handshape dataset was originally presented in [13], for the purpose of being the first dataset for handshape recognition of Argentinian Sign Language (LSA), and freely available. It consists of 16 different handshapes of the most used in the LSA. There are 10 experimental subject that made 5 different poses of each handshape, with a total of 800 images in the dataset. The interpreters had colored gloves in their hands, to facilitate the task of segment the hands in the raw image, allowing so to focus on next classification tasks. Figure 5 shows the 16 handshapes configurations presented in the dataset.

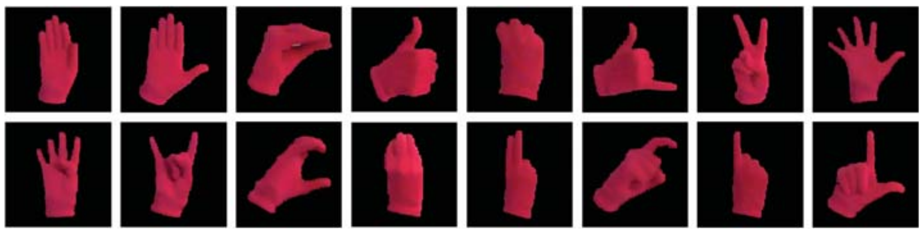

Fig. 5. Samples of the 16 handshapes of the pre-segmented version of the LSA16 dataset.

RWTH-PHOENIX-Weather (RWTH) We are employing the labeled images collected for handshape experiments in [8], which contains 3359 samples and 45 classes. We will call this dataset RWTH-PHOENIX-Weather[5] since it contains images form that dataset. Originally presented in 2012. The dataset contains video clippings of a television channel where interpreters translated reports of the state of the weather.

It is important to highlight that this dataset is highly imbalanced: some classes have over 200 examples while others have 5. In order to make a fair comparison, we removed those classes with 7 or less examples, which reduced the number of classes from 45 to 30. The resulting dataset was had 3289 examples (70 less than the original).

3.2 Comparison of models on LSA16 and RWTH

Methodology We measured the test set accuracy of the models on both datasets. Since there is no official test set for these datasets, and their size is relatively small, we averaged 100 runs of stratified randomized subsampling cross-validation to estimate the test set accuracy. We employed the Feedforward network as a baseline model, and also evaluated the LeNet, AllConvolutional, VGG, ResNet and Inception architectures.

In the case of Inception, we also employed an Inception network that was pretrained on the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC2012) dataset, and performed transfer-learning experiments by replacing the last layer by either a linear support vector machine (SVM) or a two-layer feedforward network [4]. Only these last layers were retrained, and therefore we effectively employed the output of the penultimate layer of the network as a feature.

We trained all networks using a base learning rate of 0.0007 and the ADAM learning algorithm [7], which uses a per weight adaptive learning rate to remove learning rate effects from the experiment as much as possible. The Feedforward and LeNet architectures were trained for 20 epochs and the AllConvolutional, ResNet, VGG and Inception architectures for 50, 50, 60, and 120 epochs respectively, since they contain significantly more parameters. We determined these hyperparameters via monitoring the training loss and classification error on an initial run to make sure the networks converged. We did not perform hyperparameter optimizations or modify them in subsequent experiments.

In the transfer learning experiments with the Inception network, the SVM for the final layer employed a linear kernel and a regularization coefficient $C = 0.1$. The two-layer feedforward network had a hidden layer of size 512.

Since our datasets are smaller than the standard datasets on which these models are usually evaluated, we reduced the capacity of some models to both avoid overfitting and allow the network to train succesfully. For the AllConvolutional architecture, we decreased the size of the filters to $(96, 96, 96, 192, 192, 192)$, while keeping the topology untouched. For the VGG architecture, we used the VGG16 version and set the filter sizes to $(32, 32, 64, 64, 64, 64, 64, 128, 128, 128, 128, 128, 128)$ and feedforward layer sizes to $(512, 512)$. We used the 34-layer version of the ResNet architecture, without modification [6]. The Inception network's filter's sizes were not modified in neither the transfer learning experiments nor the normal end-to-end training. Meanwhile, the LeNet architecture employed four convolutional layers with sizes $(32, 64, 128, 256)$ and 3x3 filters, and a 512-dimensional feedforward layer.

For all models, we found that adding Batch Normalization layers after each max pooling layer and replacing ReLU activation functions for ELUs reduced the training time while achieving the same accuracy. Finally, we used the presegmented RGB version of the LSA16 dataset. The code for the experiments is available at `https://github.com/midusi/convolutional_handshape`

Results In Table 1, we can observe that all models have a lower accuracy on the RWTH dataset, which is expected since it has more classes, unsegmented hand images and class imbalances. The feedforward model has the lowest accuracy in general, also expected. The convolutional architectures all have similar accuracies on each dataset. For the LSA16 dataset they achieve similar or better accuracy than ProbSom, a hand engineered method [13]. The accuracy of the DeepHand model [8] on the RWTH dataset is slightly bigger than for other models (85.50%). It must be noted that DeepHand used an Inception network pretrained on ILSVRC2014 data, along with a weakly supervised training scheme that employed 1 million example images with noisy labels and then used the RWTH dataset as testing set. No such scheme was applied with the other models. Surprisingly, the simple LeNet architecture achieved relatively good recognition rates, which perhaps is an effect of the reduced size of the datasets, which is more important for bigger models such as VGG and Inception.

Method	*LSA16*	*RWTH*
DeepHand* [8]	-	85.50
Probsom* [13]	92.30	-
Feedforward	86.58	60.27
LeNet [10]	95.78	81.19
AllConvolutional [15]	94.56	80.29
VGG16 [14]	**95.92**	**82.88**
ResNet [6]	93.49	80.89
Inception [16]	91.98	75.33
Inception+SVM [16]	93.67	78.12
Inception+NN [16]	80.62	75.97

Table 1. Accuracy of various convolutional neural network models on two datasets: LSA16 [13] and handshapes from RWTH-PHOENIX-Weather [8]. Results from methods annotated with * were taken from other papers.

3.3 Comparison of preprocessing alternatives

Methodology We also ran experiments with LeNet on LSA16 to determine whether preprocessing the images provides a boost in accuracy. We chose the LeNet because it gave good results in the previous experiments and also because since it is the simplest convolutional architecture, the results should be more representative of the general family of models. We employed the same methodology for the experiments as before. We tested the model with the raw images, and also with several variants where the background has been segmented out of the image leaving only the hand color pixels. We tested the model with the segmented image (RGB), also converting it to a grayscale image, to a black and white image, and finally calculating the contour mask of the hand.

Results Table 2 shows the accuracies obtained with the different preprocessing schemes presented and the LeNet model. From the increase in accuracy when switching from the raw RGB image to the segmented RGB image, it would

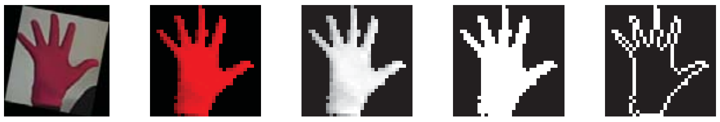

Fig. 6. From left to right: raw image, segmented image, grayscale image, hand segmentation mask, and contour.

appear that the model has trouble separating the background and the hand, and that a segmentation step helps the network to distinguish the handshapes. The Grayscale and Black and White variants seem to have a lower accuracy. This may indicate that the texture information provided by the RGB pixels helps the model (possibly because fingers can be detected more easily).

Preprocessing	Accuracy
Raw (RGB)	83.54
Segmented hand	
RGB	96.18
Grayscale	87.08
Black and White	91.38
Black and White, Contour only	80.64

Table 2. Accuracy of the LeNet model on the LSA16 dataset with different preprocessing schemes.

4 Conclusion

We have performed experiments to evaluate the mean accuracy of various prominent convolutional architectures on two handshape recognition datasets.

In the case of the LSA16 dataset, the models showed a performance on par with the state of the art. For the dataset RWTH-PHOENIX-Weather, the best architecture, VGG16, performed slightly worse than the best published method, but it is difficult to perform a direct comparison since that method employed a pretrained Inception network with a weakly-supervised learning algorithm with 1 million noisily labeled examples.

Taking that into consideration, it would appear that convolutional networks in general perform reasonably well and on the order of hand-engineered methods for the problem of handshape recognition. We also found evidence supporting the hypothesis that a pre-segmentation step can help to improve the accuracy of the recognition for these methods.

In future work, we will focus on architectures that include localization and segmentation networks as previous steps and also allow end-to-end learning.

References

1. Camgoz, N.C., Hadfield, S., Koller, O., Bowden, R.: Using convolutional 3d neural networks for user-independent continuous gesture recognition. In: Proceedings

IEEE International Conference of Pattern Recognition (ICPR), ChaLearn Workshop (2016)
2. Cooper, H., Holt, B., Bowden, R.: Sign Language Recognition, pp. 539–562. Springer London, London (2011)
3. Cooper, H., Ong, E.J., Pugeault, N., Bowden, R.: Sign language recognition using sub-units. J. Mach. Learn. Res. 13(1), 2205–2231 (Jul 2012)
4. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. CoRR abs/1310.1531 (2013)
5. Forster, J., Schmidt, C., Hoyoux, T., Koller, O., Zelle, U., Piater, J., Ney, H.: Rwth-phoenix-weather: A large vocabulary sign language recognition and translation corpus. In: Language Resources and Evaluation. pp. 3785–3789. Istanbul, Turkey (May 2012)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR abs/1412.6980 (2014)
8. Koller, O., Ney, H., Bowden, R.: Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3793–3802 (2016)
9. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553), 436–444 (2015)
10. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
11. Ronchetti, F., Quiroga, F., Estrebou, C., Lanzarini, L., Rosete, A.: Sign Languague Recognition Without Frame-Sequencing Constraints: A Proof of Concept on the Argentinian Sign Language, pp. 338–349. Springer International Publishing, Cham (2016)
12. Ronchetti, F., Quiroga, F., Estrebou, C.A., Lanzarini, L.C., Rosete, A.: Lsa64: An argentinian sign language dataset. In: XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016). pp. 794–803. Red de Universidades con Carreras en Informática (RedUNCI) (2016)
13. Ronchetti, F., Quiroga, F., Lanzarini, L., Estrebou, C.: Handshape recognition for argentinian sign language using probsom. Journal of Computer Science and Technology 16(1), 1–5 (2016)
14. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014)
15. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806 (2014)
16. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)
17. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. CoRR abs/1512.00567 (2015)
18. Von Agris, U., Zieren, J., Canzler, U., Bauer, B., Kraiss, K.F.: Recent developments in visual sign language recognition. Universal Access in the Information Society 6(4), 323–362 (2008)