# Sign-Language Recognition using Convolutional Neural Network

Anirudh Nihalani
201307695
anirudh.vattikonda@research.iiit.ac.in

Brij Mohan Lal Srivastava
201307694
brijmohanlal.s@research.iiit.ac.in

Mallikarjun BR
201307681
mallikarjun.br@research.iiit.ac.in

***Abstract*—Hand gesture recognition in real-time is a major problem in human-computer interaction and in many other fields like robotics. In this project we have attempted to recognize static hand-gestures (specifically sign-languages) by training Convolutional Neural Networks for classification of different hand-gestures using IIIT Database. In this report we will present our analysis on effect of pre-processing, model parameters and regularization techniques on the classification accuracy. We also show that the approach we have presented provides significant improvement in accuracy compared to widely used bag of visual words model.**

## I. Introduction

Gestures are recognized as crucial for human-human communication and have inspired research for human-robot interaction. Hand gestures are widely used compared to other body parts, and thus are the main focus of most of the research in this field.

The most common approach for gesture recognition is the application of feature extraction techniques to represent postures. A popular feature extraction solution is to represent the hand by matching it to a template. A problem with the template match approach is that a high variety of gestures executed by different kinds of people cannot be matched. Most of the feature extraction solutions need to segment the hand from the background of the image which can be done using a color scheme. Jmaa et al. [1] use an YCbCr color space model to separate the color information from the image luminance and obtain hand segmentation. This approach is not reliable if using a large variation in skin colors and luminance.

Deep learning models for image classification have been studied in a vast number of experiments in the past few years. Among deep learning techniques, Convolutional Neural Networks [2] have shown good results in the classification of static images [3]. The use of convolutional models focuses on how the human brain enhances and extracts features of an image in an implicit way using a set of local and global features.

We have done extensive experiments on convolutional neural networks for classification of hand gestures to understand the effect of various parameters like architectures, pre-processing and regularization on the accuracy of classification. We have used bag of visual words as our baseline classifier.

We have evaluated our techniques on two different datasets. One contains images taken from students of IIIT that contains 26 different hand gestures with varied backgrounds simulating real world scenario. The other database is a benchmark database that contains ten different hand gestures, three different backgrounds and has the hand always centered in the image.

## II. Methodology

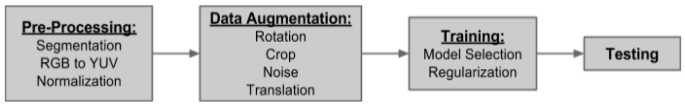


Fig. 1: Methodology

### A. Datasets

IIIT hand gesture Dataset

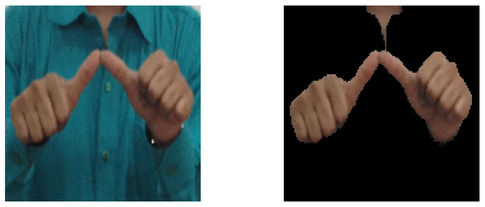
- Train Images = 1100
- Test Images = 600
- Classes = 26

Triesche Dataset

- Train Images = 600
- Test Images = 180
- Classes = 10

### B. Pre-Processing

*1) Segmentation:* The IIIT dataset contains images covering entire person but since we are interested in classification of hand gestures we have to segment the images to have just the hands as other portion of the image will be irrelevant and in most cases misleading for our task.



(a) Unsegmented Image (b) Segmented Image

Fig. 2: Segmentation

*2) RGB to YUV:* For the task of hand gesture recognition intensity variation is a better discriminative feature than chrominance. Since YUV images represent such intensity variation better than RGB images we have converted the images to YUV.

*3) Normalization:* If the dynamic range of one channel is very large while the dynamic range of other channels is relatively very low then the training will bias towards the larger dynamic range channels. In order to avoid such bias we have normalized the images per channel to the same range.

## *C. Data Augmentation*

Since the number of training images are very less the classifier overfits training data very soon. So we have augmented the existing data using combination of 10 clockwise and counter-clockwise rotation, 5% translation, addition of Gaussian and Salt & Pepper noise and cropping, thereby increasing the training dataset size to 33, 000 images. By including such augmentations we hope that the trained model would be more robust to various real world effects like angle of camera, occlusions etc.
The easiest and most common method to reduce overfitting on image data is to artificially enlarge the dataset using label-preserving transformations. We employ four distinct forms of data augmentation, all of which allow transformed images to be produced from the original images with very little computation.

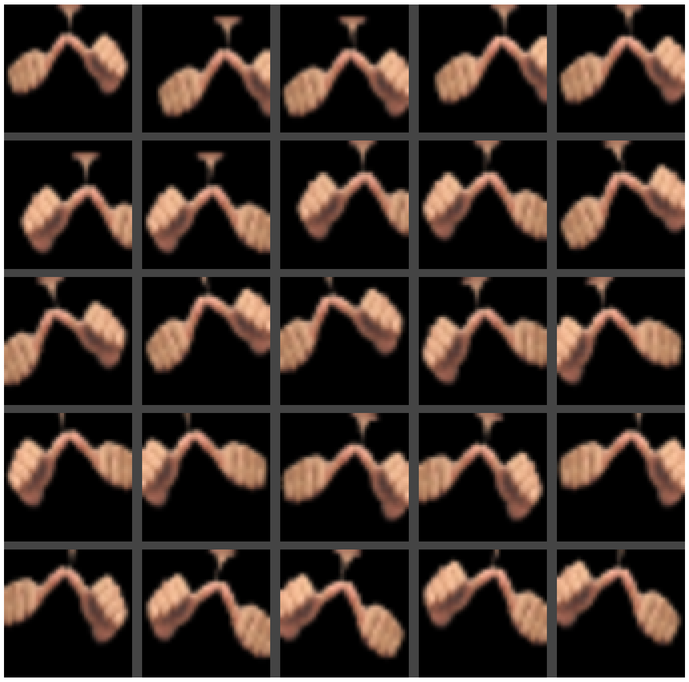


Fig. 3: Augmented data

The form of data augmentation consists of generating image translations, rotations, scaling and noise. This increases the size of our training set by a factor of 30, though the resulting training examples are, of course, highly inter-dependent. Without this scheme, our network suffers from substantial overfitting, which would have forced us to use much smaller networks.

## *D. Training*

In order to train a Convolutional neural network there a lot of parameters which can be optimized/tweaked based on the task to improve the performance. We have performed a lot of experiments training CNN by varying all such parameters like Convolution Kernel size, Pool size, Type of pooling, Image resolution, Optimization Algorithm, depth and width of convolutions, depth of MLP, type of activation functions, Regularization techniques like Dropout, Weight Decay and Normalization etc.

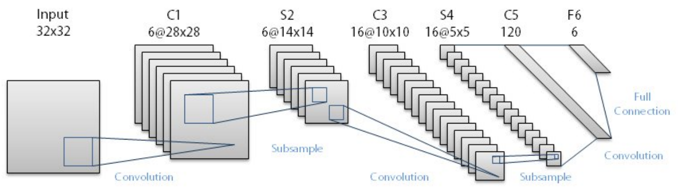


Fig. 4: CNN Architecture

*1) Convolutional Neural Network:* Convolutional networks combine three architectural ideas to ensure some degree of shift and distortion invariance: local receptive elds, shared weights (or weight replication), and, sometimes, spatial or temporal subsampling. A typical convolutional network is shown in figure 4. The input plane receives images of characters that are approximately size-normalized and centered. Each unit of a layer receives inputs from a set of units located in a small neighborhood in the previous layer.

The idea of connecting units to local receptive fields on the input goes back to the Perceptron in the early 60s, and was almost simultaneous with Hubel and Wiesel's discovery of locally-sensitive, orientation-selective neurons in the cat's visual system. Local connections have been reused many times in neural models of visual learning. With local receptive elds, neurons can extract elementary visual features such as oriented edges, end-points, corners (or similar features in speech spectrograms). These features are then combined by the higher layers. As stated earlier, distortions or shifts of the input can cause the position of salient features to vary.

In addition, elementary feature detectors that are useful on one part of the image are likely to be useful across the entire image. This knowledge can be applied by forcing a set of units, whose receptive elds are located at different places on the image, to have identical weight vectors (Rumelhart, Hinton and Williams, 1986). The outputs of such a set of neurons constitute a feature map . At each position, different types of units in different feature maps compute different types of features.

A sequential implementation of this, for each feature map, would be to scan the input image with a single neuron that has a local receptive field, and to store the states of this neuron at corresponding locations in the feature map. This operation is equivalent to a convolution with a small size kernel, followed by a squashing function. The process can be performed in parallel by implementing the feature map as a plane of neurons that share a single weight vector. Units in a feature map are constrained to perform the same operation on different parts of the image. A convolutional layer is usually composed of several feature maps (with different weight vectors), so that multiple features can be extracted at each location.

The first hidden layer in figure 4 has 6 feature maps with 5 by 5 receptive fields. Shifting the input of a convolutional layer will shift the output, but will leave it unchanged otherwise. Once a feature has been detected, its exact location becomes less important, as long as its approximate position relative to other features is preserved. Therefore, each convolutional layer is followed by an additional layer which performs a local averaging and a subsampling, reducing the resolution of the feature map, and reducing the sensitivity of the output to shifts and distortions. The second hidden layer in figure 4 performs 2 by 2 averaging and subsampling, followed by a trainable coefficient, a trainable bias, and a sigmoid. The trainable coefficient and bias control the effect of the squashing non-linearity (for example, if the coeffcient is small, then the neuron operates in a quasi-linear mode).

Successive layers of convolutions and subsampling are typically alternated, resulting in a "bi-pyramid" at each layer, the number of feature maps is increased as the spatial resolution is decreased. Each unit in the third hidden layer in figure 4 may have input connections from several feature maps in the previous layer.

Since all the weights are learned with back-propagation, convolutional networks can be seen as synthesizing their own feature extractor. The weight sharing technique has the interesting side effect of reducing the number of free parameters, thereby reducing the "capacity" of the machine and improving its generalization ability (see (LeCun, 1989) on weight sharing, and learning and generalization for an explanation of notions of capacity and generalization). The network in figure 4 contains about 100,000 connections, but only about 2,600 free parameters because of the weight sharing. Such networks compare favorably with other methods on handwritten character recognition tasks (Bottou et al., 1994), and they have been deployed in commercial applications.

*2) Dropout:* Combining the predictions of many different models is a very successful way to reduce test errors, but it appears to be too expensive for big neural networks that already take several days to train. There is, however, a very efficient version of model combination that only costs about a factor of two during training. The recently-introduced technique, called dropout [4], consists of setting to zero the output of each hidden neuron with probability 0.5. The neurons which are dropped out in this way do not contribute to the forward pass and do not participate in back-propagation. So every time an input is presented, the neural network samples a different architecture,but all these architectures share weights. This technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons. It is, therefore, forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons. At test time, we use all the neurons but multiply their outputs by 0.5, which is a reasonable approximation to taking the geometric mean of the predictive distributions produced by the exponentially-many dropout networks.

We use dropout in all the convolution layers. We did not notice any significant change either in classification accuracy or in speed of convergence by using dropout.

*3) Non-Linearities and Weight Normalization:* In traditional ConvNets this simply consists in a pointwise $tanh()$ sigmoid function applied to each site $(ijk)$. However, recent implementations have used more sophisticated non-linearities. A useful one for natural image recognition is the rectified sigmoid or $ReLU$:

$$R_{abs} : abs(g_i.tanh())$$

where $g_i$ is a trainable gain parameter. The rectified sigmoid is sometimes followed by a subtractive and divisive local normal-ization $N$, which enforces local competition between adjacent features in a feature map, and between features at the same spatial location. The subtractive normalization operation for a given site $x_{ijk}$ computes:

$$v_{ijk} = x_{ijk} \sum_{ipq} w_{pq}.x_{i,j+p,k+q} \qquad ,$$

where $w_{pq}$ is a normalized truncated Gaussian weighting window (typically of size 9x9). The divisive normalization computes

$$y_{ijk} = v_{ijk}/max(mean(\sigma_{jk}), \sigma_{jk})$$

where $\sigma_{jk} = (\sum_{ipq} w_{pq}.v_{i,j+p,k+q}^2)^{\frac{1}{2}}$. The local contrast normalization layer is inspired by visual neuroscience models.

We observed that in our case $tanh()$ converges faster than $ReLU$. We also employed Local Contrastive Normalization for weights which slightly improves the classification accuracy.

*4) Feature Pooling Layer:* This layer treats each feature map separately. In its simplest instance, called $P_A$, it computes the average values over a neighborhood in each feature map. The neighborhoods are stepped by a stride larger than 1 (but smaller than or equal the pooling neighborhood). This results in a reduced-resolution output feature map which is robust to small variations in the location of features in the previous layer. The average operation is sometimes replacedby a max $P_M$. Traditional ConvNets use a pointwise $tanh()$ after the pooling layer, but more recent models do not. Some ConvNets dispense with the separate pooling layer entirely , but use strides larger than one in the filter bank layer to reducethe resolution. In some recent versions of ConvNets ,the pooling also pools similar feature at the same location, in addition to the same feature at nearby locations. Supervised training is performed using a form of stochastic gradient descent to minimize the discrepancy between the desired output and the actual output of the network. All the coefficient of all the filters in all the layers are updated simultaneously by the learning procedure. The gradients are computed with the back-propagation method.

### *E. Testing*

To avoid overfitting we have used the classification accuracy on test data after each epoch as the termination criterion.

## III. Experiments and Results

| S.no | Data | Filter Size | Pool Size | Layers | Train | Test |
|---|---|---|---|---|---|---|
| 1 | 1046x3x128x128 | 5x5 | 2x2 | yuv(3x128x128)->16x62x62->256x29x29->2-layered MLP(128->26) | 100% | 56% |
| 2 | 5230x3x128x128 | 5x5 | 2x2 | yuv(3x128x128)->16x62x62->256x29x29->2-layered MLP(128->26) | 99% | 55% |
| 3 | 5230x3x128x128 | 31x31 | 2x2 | yuv(3x128x128)->16x49x49->256x9x9->2-layered MLP(128->26) | 87% | 46% |
| 4 | 33000x3x64x64 | 5x5 | 2x2 | yuv(3x64x64)->16x30x30->64x13x13->256x4x4->2-layered MLP(128->26) | 99% | 63% |

TABLE I: Experiments and Results on IIIT Database

| S.no | Data | Filter Size | Pool Size | Layers | Train | Test |
|---|---|---|---|---|---|---|
| 1 | 75x1x64x64 | 5x5 | 2x2 | 1x64x64 ->16x30x30 ->64x13x13 ->256x4x4 ->128 ->10 | 100% | 76.6% |
| 2 | 75x1x128x128 | 5x5 | 3x3 | 1x128x128 ->16x41x41 ->64x12x12 ->256x2x2 ->128 ->10 | 99% | 80% |
| 3 | 75x1x128x128 with Sobel | 5x5 | 2x2 | 1x128x128 ->16x41x41 ->64x12x12 ->256x2x2 ->128 ->10 | 100% | 68% |
| 4 | 75x1x128x128 + Sobel + Dropout + ReLU | 5x5 | 2x2 | 1x128x128 ->16x41x41 ->64x12x12 ->256x2x2 ->128 ->10 | 99% | 63% |
| 5 | 75x1x128x128 +Sobel +tanh +Normalization | 5x5 | 2x2 | 1x128x128 ->16x41x41 ->64x12x12 ->256x2x2 ->128 ->10 | 100% | 73.33% |

TABLE II: Experiments and Results on Triesch Database

### A. *Baseline Model*

Similar to terms in a text document, an image has local interest points or key points defined as salient image patches (small regions) that contain rich local information of the image. Keypoints are usually around the corners and edges of image objects, such as the edges of the map and around peoples faces. We use the Difference of Gaussian (DoG) detector [5] to automatically detect keypoints from images. The detected keypoints are depicted using PCA-SIFT descriptor, which is a 36-dimensional real-valued feature vector. An image can be represented by a set of keypoint descriptors, but this set varies in cardinality and lacks meaningful ordering. This creates difficulties for many learning methods (e.g., supervised classifiers) that require feature vectors of fixed dimension as input. To address this problem, we use the vector quantization (VQ) technique which clusters the keypoint descriptors in their feature space into a large number of clusters using the K-means clustering algorithm and encodes each keypoint by the index of the cluster to which it belongs. We conceive each cluster as a visual word that represents a specific local pattern shared by the keypoints in that cluster. Thus, the clustering process generates a visual-word vocabulary describing different local patterns in images. By mapping the key points to visual words, we can represent each image as a bag of visual words. This representation is analogous to the bag-of-words document representation in terms of form and semantics. Both representations are sparse and high-dimensional, and just as words convey meanings of a document, visual words reveal local patterns characteristic of the whole image.

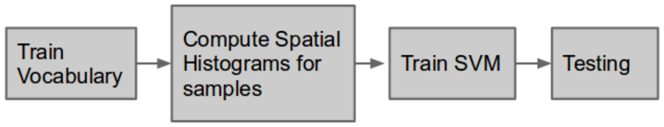


Fig. 5: Baseline Model

SVM is used as the classifier for the different classes that are trained based on Bag of Visual words model.

## IV. Conclusion

Based on the analysis of various experiments we can conclude that data preprocessing,data augmentation helped in improving the classificaiton accuracy. Model parameters such as depth of convolution, number of feature maps per layer,Convolution Kernel size, Pool size need to be adjusted such that each unit in the last convolution layer would have a receptive field of atleast 80% of the input image. Regularization techniques such as Dropout, Weight Decay has only improved the classification accuracy by negligible factors. Compared with the baseline model the accuracy has improved by two folds by using CNN for classfication. Although CNN gave very good results for some classes the model gave mixed results for similar looking hand gestures like for M and N.

## Acknowledgment

The authors would like to thank Ridhiman Dasgupta for the insightful discussions and for his help in providing infrastructure for running experiments.

## References

[1] Jmaa, A.B., Mahdi, W., Jemaa, Y.B., Hamadou, A.B.: Hand localization and fingers features extraction: Application to digit recognition in sign language. In: Corchado, E., Yin, H. (eds.) IDEAL 2009. LNCS, vol. 5788, pp. 151159. Springer, Heidelberg (2009)

[2] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE, 22782324 (1998)

[3] Nagi, J., Ducatelle, F., Di Caro, G., Ciresan, D., Meier, U., Giusti, A., Nagi, F., Schmidhuber, J., Gambardella, L.: Max-pooling convolutional neural networks for vision-based hand gesture recognition. In: IEEE International Conference on Signal and Image Processing Applications (ICSIPA), pp. 342347 (2011)

[4] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov. Improving neural net- works by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 , 2012.

[5] D. G. Lowe. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision, 60(2):91110, 2004.