CS230

Sign Language Translation Using Ladder Networks

Guanzhou Ye
Department of Computer Science
Stanford University
markye@stanford.edu

Abstract

I translate sign language images into text while using significantly fewer labelled data than traditional models. I use a convolutional ladder network and demonstrate how it can perform this task by achieving superior performance while using 80% less data when compared to traditional convolutional neural networks.

1 Introduction

My project aims to demonstrate how ladder networks can translate sign language images into text while using significantly fewer labelled data than traditional models. There is a lot of unlabelled sign language data available (e.g. recording sign language conversations; videos of sign language interpreters at public events), but it is time consuming to label this data because each word in the English dictionary requires a separate label. A ladder network can greatly reduce the amount of labels required and make this task more feasible.

The model is a ladder network (a special type of neural network, explained below). The input will be images of hand signs and the output will be a one-hot encoded vector specifying the translated word.

2 Related work

There have been prior work on translating sign language images to text. Vivek Bheda and N. Dianna Radpour used CNNs to convert sign language images to text [1]. They used a dataset with about 67 images per class [2] to achieve 82.5% accuracy [1]. This can be improved on in both accuracy (since both human and Bayes error are close to 100% accuracy) and reduced number of labels. My method differs from this approach because I use fewer labels - traditional labelling for sign language translation takes too much time because each word requires multiple labelled data.

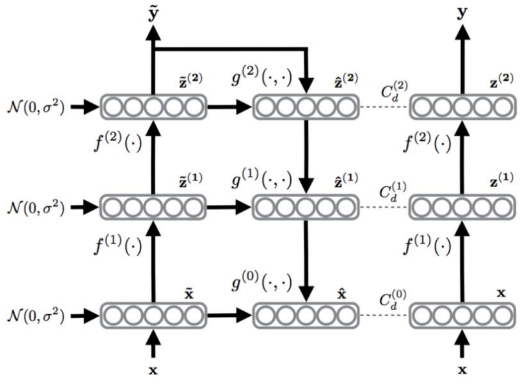
There has been prior work on using ladder networks for classification using minimal labelled data. Ladder networks have been used for classifying handwritten digits (achieved 99.11% using only 10 labels per class) [3], human activity classification [4], and sequence models [5]. My methodology differs from these in my application (sign language translation) and hyperparameter choices.

3 Dataset and Features

I am using a Sign Language dataset from Kaggle [6] as my experimental dataset. The dataset contains the alphabet except for J and Z, as they are signed using motions (out of scope for this project, but hopefully the principles established here can be used in a future motion-capturing RNN model). In total, there are 23 classes.

Figure 1: Sample images from the dataset [6].

Figure 2: Ladder network architecture, from the original paper [3].



My dataset contains 27455 training images, 3586 validation images, and 3586 test images. Each input image is 28x28 pixels and is already greyscaled. Each pixel became an input feature. I performed preprocessing by linearly normalizing the images' pixel values to be between 0 and 1. I did not perform image augmentation because I had sufficient data and because of my goal of using fewer labelled images.

The distribution of each class is roughly even - there is an average of 1143.96 images per class with a standard deviation of 81.9951620355 and a max/min of 1294 and 957 respectively.

Even though this dataset is labelled, I ignored most of these labels because my goal is to use as little labelled data as possible. Instead, the ladder network I used relied on semi-supervised learning techniques.

4 Methods

4.1 Ladder Network Overview

I used a ladder network. Ladder networks perform semi-supervised learning by combining a traditional DNN (input is features, output is the predicted label) and an autoencoder (reconstructs activations at each layer) [3].

Here is an explanation of how it works. Please see Figure 2 for reference.

1. x is the input and y is the output (i.e. the predicted value).
2. x is fed into the regular DNN (right) - the layers and values from this DNN are called "clean".

Figure 3: Cost function of the ladder network (retrieved from [7]).

$$Cost = -\Sigma_{n=1}^{N} \log P\big(\tilde{y}(n) = y^*(n)|x(n)\big) +$$
$$\Sigma_{n=N+1}^{M} \Sigma_{l=1}^{L} \lambda_l \, \mathbf{ReconsCost}(z^{(l)}(n), \hat{z}^{(l)}(n))$$

3. On the left, x is fed into the autoencoder. The autoencoder uses the same layers and weights as the DNN, except Gaussian noise is added to each layer's inputs. This noise serves as a regularizing effect.
4. The autoencoder then propagates backwards and attempts to recreate each layer's clean "z" values (i.e. after dot product with weights but before activation) using a denoising function, the corrupted z value, and the previous layer's (i.e. closer to y) z value.

4.2 Cost Function

The loss function is the combination of the classification loss (cross entropy loss from the encoding DNN) and the sum of all the layers' "reconstruction cost" (i.e. difference between denoised z value and clean z value). For convolutional ladder networks, only the last layer's reconstruction cost is used [1].

Image source: [7] Mohammad Pezeshki, Linxi Fan, Philemon Brakel, Aaron Courville, Yoshua Bengio. Deconstructing the Ladder Network Architecture. arXiv preprint arXiv:1511.06430, 2015.

The reconstruction cost forces the network to learn a set of weights that result in similar entities also having similar "z" values. The loss will be high if slight changes in one layer's "z" values causes vast differences in other layers "z" values. Therefore, similar entities have similar "z" values and therefore similar predictions from the DNN. Subsequently, data that hasn't been seen by the model will be accurately classified because their "z" values will be similar to that of labelled data, and they will be classified accordingly.

By combining the cross entropy loss and reconstruction loss, the ladder network learns weight distributions such that it generates accurate predictions AND can generalize to unseen data.

I will use convolutional layers within my ladder network. The convolutional layers are more suitable for image classification because they allow a filter to be used in multiple locations on the image and are less susceptible to noise. Convolutional ladder networks are similar to regular ladder networks except their reconstruction cost is only comprised of the last layer's reconstruction cost [1].

5 Experiments/Results/Discussion

5.1 Experiment Setup

For my experiment, I trained a baseline CNN model and my ladder network using different numbers of labelled examples. I used these results to demonstrate the ladder network can achieve superior accuracy with fewer labelled examples.

My primary metric is accuracy. I chose accuracy because it is the best measure of whether we are translating signs correctly. We don't need to worry about imbalanced classes inflating the accuracy because the classes are roughly balanced.

5.1.1 Model Architecture

I'm using a convolutional ladder network with the following architecture: 32 3x3 filters, 64 3x3 filters, 64 3x3 filters, 128 3x3 filters, 128 fully connected layer, and a softmax layer.

Finding this optimal architecture was an iterative process. Here is how I did it:

1. I started with a fully connected ladder network (no convolutional layers), but the variance was too high. This indicated my model was overfitting. Using convolutional layers reduced overfitting because they are less susceptible to minor feature skews and offsets.

2. I faced an underfitting problem when I first tried using convolutional layers, as evidenced by a high avoidable bias. I originally used 16, 32, and 64 3x3 filters in a 3 layer configuration. I added more filters and changed the fully connected architecture to be 2 layers (500, 250).
3. The previous step led to overfitting as the variance was high. I solved this by changing the fully connected layers to a single layer with 128 nodes.

5.1.2 Hyperparameter Tuning

The first step was to tune hyperparameters for an optimal model. I used the coarse-to-fine process we learned in class. I found the following to be optimal:

Batch size: 32 Learning rate: 0.001 Denoising cost for last layer (multiplied with reconstruction cost): 3 Noise standard deviation: 0.05

5.2 Results

These results were evaluated using the test dataset.

Table 1: Results

Model Architecture	Num Labelled Per Class	Accuracy	Precision	Recall
Baseline CNN	10	0.5297	0.5313	0.5114
Ladder Network	10	0.6606	0.6382	0.6481
Baseline CNN	40	0.7379	0.7573	0.7496
Ladder Network	40	0.8447	0.7874	0.8021
Baseline CNN	70	0.7799	0.7666	0.7679
Ladder Network	70	0.8898	0.8256	0.8436
Baseline CNN	100	0.8118	0.8030	0.8000
Ladder Network	100	0.9274	0.8801	0.8940
Baseline CNN	200	0.8302	0.8190	0.8134
Ladder Network	200	0.9533	0.9048	0.9166

The convolutional ladder network outperformed the baseline CNN model in all situations. This is expected, given that the ladder network includes a reconstruction cost that allows it to better generalize to all data.

The ladder network required only 40 labels to achieve a higher accuracy than the CNN did using 200 labels, which is a reduction of 80%.

5.3 Error Analysis

The model is overfitting but this is expected because I am deliberately using as little data as possible. As such, the model doesn't have enough data to generalize to unseen cases very well. In all the above cases, training accuracy was about 98%. The variance is therefore quite high for most cases (except for the model trained on 200 labelled examples) because the difference between training and validation error is many times the difference between training and human error. This combined with the high training accuracy indicates overfitting.

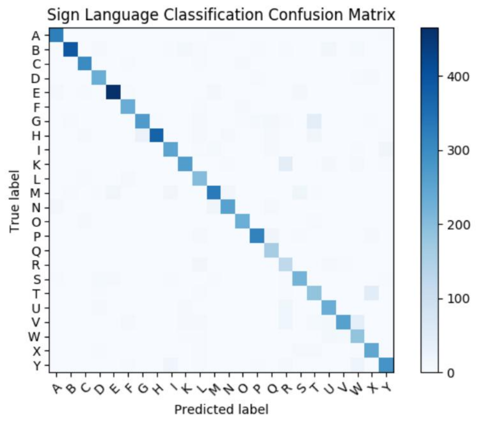
There is also some avoidable bias. Human error and Bayes error should be very similar to each other and both close to 0 since sign language recognition is easily performed by humans with a high degree of accuracy. The small amount of avoidable bias (about 2%, since training accuracy is about 98%) is expected since ladder networks add Gaussian noise into each layer.

5.3.1 Confusion Matrices

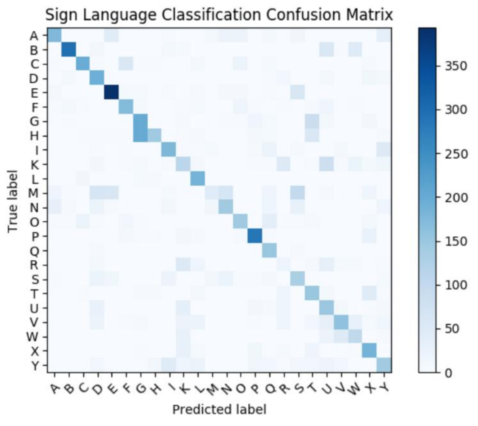
The confusion matrices below indicates which classes the model is likely to mispredict, and what the mispredicted class is. I analyzed 2 confusion matrices: first one for the model trained using 200 labelled examples and the second one for the model trained using 10 labelled examples.

Figure 4b. has a higher rate of mispredictions because the model uses less data and therefore generalizes much less.

Figure 4: Confusion Matrices



(a) 200 labels per class



(b) 10 labels per class

Figure 5: G is often misclassified as T, which isn't surprising since they both involve a protruding finger from a fist. It's also often misclassified as an H, since they are almost identical (except for one finger). T and H are different enough from each other such that we don't see many misclassifications between them.

(a) Sign for G

(b) Sign for T

(c) Sign for H

5.3.2 Common Mispredictions

Using the confusion matrix, I looked at some of the most common classes of misprediction. As expected, signs that look very similar are more likely to be misclassified with each other because the model doesn't have enough data to learn the details that can generalize for distinguishing between the two classes.

6 Conclusion/Future Work

In this project, I demonstrated how a convolutional ladder network can achieve 25% higher accuracy than traditional convolutional neural networks in situations when there isn't a lot of labelled data. The ladder network required 80% less data to exceed the performance of a baseline CNN - this greatly reduces the amount of labels required.

This concept can be extended to other image recognition problems as well. it is difficult to obtain labelled data. Reducing the amount of labelled data required can greatly assist in model training.

Given more time, I would first like to expand the number of words the sign language translation model supports. I would also like to explore using an RNN ladder network for translating sign language videos. Given even more time, I would productionize this system by building in image localization to identify signs in a larger image and classify them in real time.

7 Contributions

I did all the work (model training, writeup, poster, etc.) as the only person on the team.

The code repository can be found here: https://github.com/mark-g-y/SignLanguageLadderNetwork

References

[1] Vivek Bheda and N. Dianna Radpour. Using Deep Convolutional Networks for Gesture Recognition in American Sign Language. arXiv preprint arXiv:1710.06836, 2017.

[2] Barczak, A.L.C., Reyes, N.H., Abastillas, M., Piccio, A., Susnjak, T. A new 2D static hand gesture colour image dataset for ASL gestures, Research Letters in the Information and Mathematical Sciences, 15, 12-20, https://mro.massey.ac.nz/handle/10179/4514, 2011.

[3] Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund. Semi-Supervised Learning with Ladder Networks. arXiv preprint arXiv:1507.02672, 2015.

[4] Ming Zeng, Tong Yu, Xiao Wang, Le T. Nguyen, Ole J. Mengshoel, Ian Lane. Semi-Supervised Convolutional Neural Networks for Human Activity Recognition. arXiv preprint arXiv:1801.07827, 2018.

[5] Isabeau Prémont-Schwarz, Alexander Ilin, Tele Hotloo Hao, Antti Rasmus, Rinu Boney, Harri Valpola. Recurrent Ladder Networks. arXiv preprint arXiv:1707.09219, 2017.

[6] tecperson. Sign Language MNIST (Version 1) [Drop-In Replacement for MNIST for Hand Gesture Recognition Tasks]. 2018.

[7] Mohammad Pezeshki, Linxi Fan, Philemon Brakel, Aaron Courville, Yoshua Bengio. Deconstructing the Ladder Network Architecture. arXiv preprint arXiv:1511.06430, 2015.

Code References:

[1] divamgupta. Semi-Supervised Learning with Ladder Networks in Keras. Github, https://github.com/divamgupta/ladder_network_keras, 2019.

[2] Confusion Matrix. Sci-kit Learn, https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html.