Danmarks Tekniske Universitet

Andreas Nilausen, s183915
August Semrau Andersen, s183918
William Marstrand, s183921

Introduction to Intelligent Systems 02461

ASL CLASSIFICATION

Number of characters: 12600 , Standard Pages: 5.25

January 23, 2019, Danmarks Tekniske Universitet.

Abstract

Motivated by the intriguing challenge of recognizing American Sign Language (ASL) letters and what benefits this technology could have, this study set out to investigate the implementation of a Convolutional Neural Network (CNN) for image recognition. Inspired by a state of the art CNN made by StradigiAI, we approached the challenge with a desire to find out if the specific setup used in StradigiAI's CNN was superior to other setups available in regards to which pooling functions and optimization algorithms were used. Using bootstrapping as a statistical tool, an experiment aiming to test the influence of different independent variables on the accuracy of the model was performed and for this experiment it was found that changing the pooling functions and optimization algorithms had no significant effect on the accuracy of the model. As no one model could be determined as being superior to the other, the base model was chosen for the final phase of the study in which the CNN model was trained on a large scale dataset[1] of the ASL letters A to E, resulting in a CNN with 71% validation accuracy.

[1]15000 training images and 620 test images.

1 Introduction

Recognizing sign language proposes an interesting challenge due to the multiple factors involved, like similarity in the shapes of the signs and changing angles of the hands etc.. The complexity is increased further when working with a 2D-input as no depth can be used to tell the different gestures apart. This make it an interesting and challenging case to work with.

The company StradigiAI created a Convolutional Neural Network (CNN) for multi-label classification through supervised learning to solve exactly this task and showed great results at NeurIPS 2018 [5]. As of now, they have not published the exact code for the model or the data used to train it, only a surface scratching representation of their model is available.

1.1 Research Question

This study investigates the effects of tuning different independent variables in a CNN with the same setup in regards to layer types as that of StradigiAI. A set of different setups are tested in individual models and compared on accuracy using confidence intervals, and a final model is chosen for categorizing the letters A to E from the ASL alphabet.

1.2 Hypothesis

Through the study we expect to find a combination of independent variables that individually increases accuracy and combined will produce a model with higher accuracy than the individually tested models.

2 Methods

This experiment was designed for comparing different CNN's in their ability to recognize the letters A, B, C, D and E from ASL using supervised learning. Below, the independent variables are listed as well as the variable that is dependent on the influence of these, ie. the dependent variable.

Independent Variables	**Dependent Variable**	
Max Pooling vs Average Pooling	Accuracy in ASL Letter Recognition	(1)
Augmentation vs No Augmentation		
SGD vs Adadelta vs Adam		

The initial network layer architecture for this study was inspired by a CNN created by StradigiAI for NeurIPS 2018[5]. The network was build using the Keras API.

2.1 Equipment and Software

Equipment

- Laptop: MSI GE63VR 7RE Raider with GTX 1060 GPU. (Training and testing model)
- Webcam: HD type, 30fps@720p (Live predictions)
- Camera: Panasonic Lumix DMC-FT30EB-D (For expanding training and testing image datasets)

Software

- Python v3.6.8 on Windows 10 64 bit OS
- Keras API v2.2.4 on top of Tensorflow-GPU v1.12.0
- VLC Media Player v3.0.6

2.2 Pilot Studies

Pilot studies were conducted to investigate if the public data sets available would be useful for training the CNN and whether it would be feasible to apply the CNN on all 26 ASL letters.
Results from training the network on public data from Kaggle[2] lead to the conclusion that the publicly available datasets were of too poor quality and a new set of images had to be produced for the experiment.
Training the CNN on the entire ASL alphabet proved too complex and required a very large amount of high quality data. With this in mind it was deemed not feasible to do the entire alphabet compared to a smaller part of it, and for this reason a subset of five letters A to E were chosen for the experiment of this study.

2.3 Experimental Procedure

The procedure was designed as an iterative process inspired by the agile software development approach[1]. The process consisted of 3 phases as illustrated in the figure below: measuring performance of multiple trained models, choosing models, and composing a new model with the features from the chosen models.

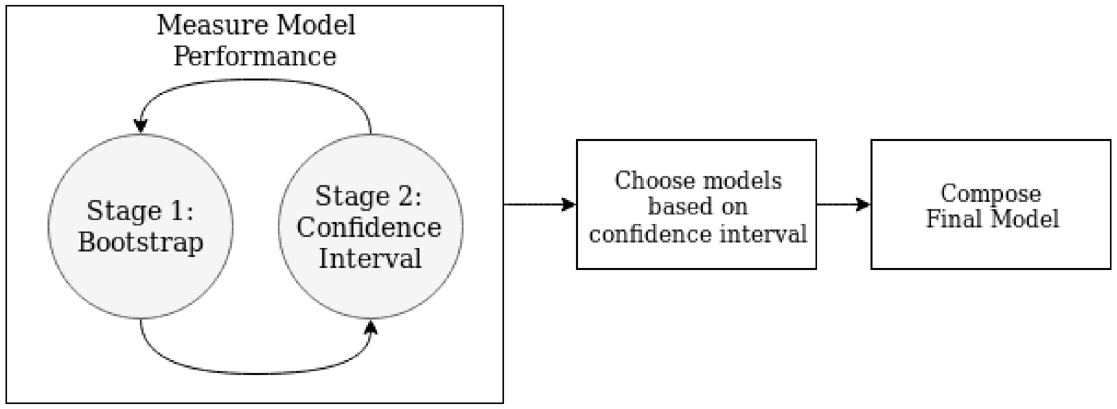

Measure Model Performance
Stage 1: Bootstrap
Stage 2: Confidence Interval
Choose models based on confidence interval
Compose Final Model

Figure 1: Experimental Procedure

A basis for the models in the measurement-performance phase was chosen to have fixed attributes from which the independent variables in separate tests were implemented to be statistically evaluated.

The basis had the following attributes:

- Uses basic augmentation[2]
- Utilizing MaxPooling2D.
- Optimization algorithm: SGD (Stochastic Gradient Descent) with default Keras settings.[3]

[2]rescale=1./255, shear_range=0.2, rotation_range=25, zoom_range=0.2, zoom_range=0.2, height_shift_range=0.1, horizontal_flip=True, fill_mode='nearest'.

[3]learning rate=0.01, momentum=0, decay=0, nesterov=False.

Four different models were tested by changing the following attributes - all of which independent variables - in the base-model.

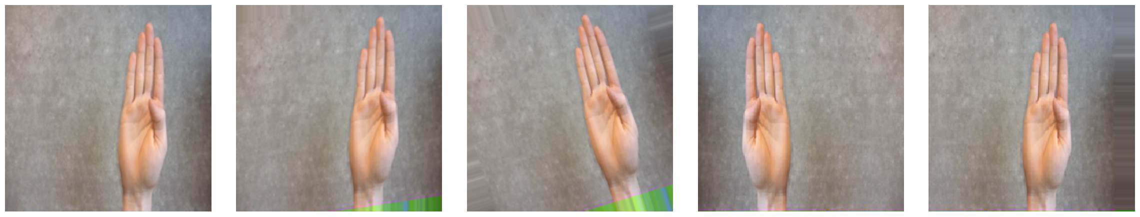
1. Augmentation $\rightarrow$ No augmentation.
2. MaxPooling2D $\rightarrow$ AveragePooling2D.
3. SGD $\rightarrow$ 'Adadel' with default Keras settings[4].
4. SGD $\rightarrow$ 'Adam' with default Keras settings[5].

2.4 Data Collection

A dataset of images were created by filming video sequences of hands showing the ASL letters and thereafter splitting these sequences up in separate frames using VLC Media Player. This resulted in a dataset of 150 images for each letter A to E, each with 200x200 resolution. The small datasets were used to make the measurement process in the first phase shorter, but more importantly to enable the tests to be run without overloading the working memory (RAM).

2.5 Data Preprocessing

As the chosen method of data collection caused a high degree of invariability in the dataset, image augmentation was used to establish variance in the training data. By augmenting the training images, overfitting is prevented[6] through reducing the effect of frequently appearing features such as recurrent hand positioning. By analyzing the pilot studies, augmentation-methods for the experiment were chosen and parameters were optimized.



(a) Original. (b) Shearing. (c) Rotation. (d) Flipping. (e) Shifting.

Figure 2: Examples of Different Augmentation Methods

[4]lr=1.0, rho=0.95, epsilon=None, decay=0.0
[5]lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False

2.6 Layer Structure

The base-model structure of the CNN for all tested models were inspired by the setup StradigiAI[5] runs in their CNN. The structure is visualized below.

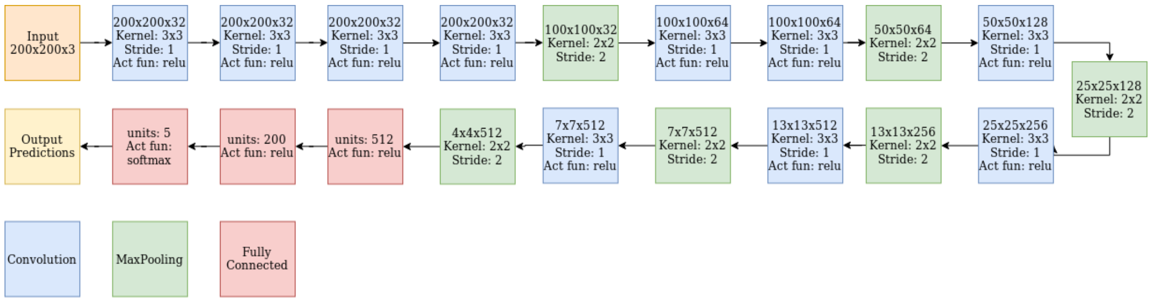

Input 200x200x3
200x200x32 Kernel: 3x3 Stride: 1 Act fun: relu
200x200x32 Kernel: 3x3 Stride: 1 Act fun: relu
200x200x32 Kernel: 3x3 Stride: 1 Act fun: relu
200x200x32 Kernel: 3x3 Stride: 1 Act fun: relu
100x100x32 Kernel: 2x2 Stride: 2
100x100x64 Kernel: 3x3 Stride: 1 Act fun: relu
100x100x64 Kernel: 3x3 Stride: 1 Act fun: relu
50x50x64 Kernel: 2x2 Stride: 2
50x50x128 Kernel: 3x3 Stride: 1 Act fun: relu
25x25x128 Kernel: 2x2 Stride: 2
25x25x256 Kernel: 3x3 Stride: 1 Act fun: relu
13x13x256 Kernel: 2x2 Stride: 2
13x13x512 Kernel: 3x3 Stride: 1 Act fun: relu
7x7x512 Kernel: 2x2 Stride: 2
7x7x512 Kernel: 3x3 Stride: 1 Act fun: relu
4x4x512 Kernel: 2x2 Stride: 2
units: 512 Act fun: relu
units: 200 Act fun: relu
units: 5 Act fun: softmax
Output Predictions
Convolution
MaxPooling
Fully Connected

Figure 3: Our Convolutional Neural Network Structure.

2.7 Evaluation of Training Models

As Deep Learning models are stochastic and therefore uses randomization to initialize the model weights, a bootstrapping[8] technique inspired by articles by Jason Brownlee (PhD in software engineering), was chosen to evaluate the performance of the different models[3].
In this process, k number of bootstrap iterations with 80% training and 20% test split were run. For each iteration the training set of images were chosen with replacement, and the validation set became the remaining images not in the training set. A model, with randomly initialized weights, for each bootstrap iteration was then fit to the training data and the accuracy on predicting the labels of test images was registered.

2.8 Methods for Analyzing Results

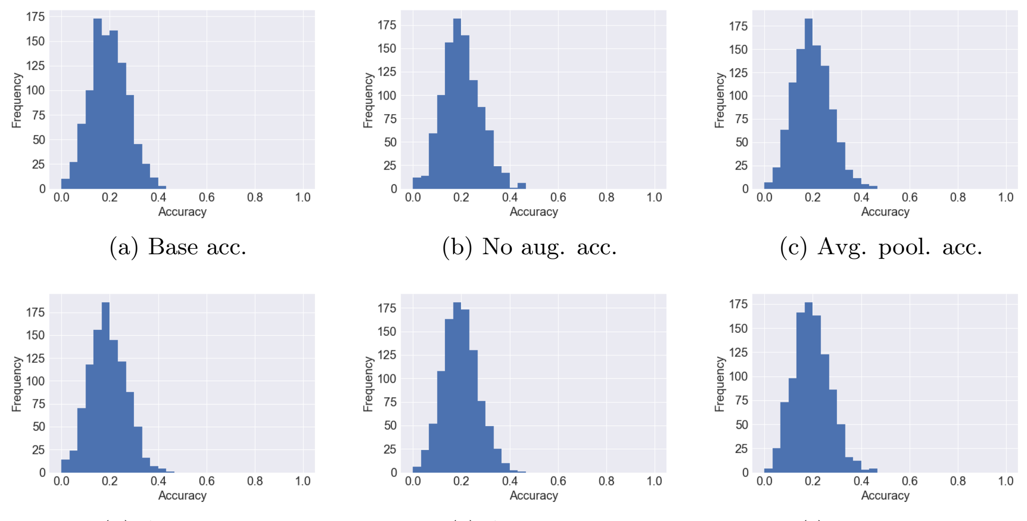
Using the bootstrapping accuracy scores it was possible to determine normal distributions for the accuracies of the trained models. Using these scores, accuracy confidence intervals were calculated and compared to see which models had the highest lower bound and are thereby most fit for having their variable settings combined into a final model.

A way of interpreting the confidence interval for the accuracy of a CNN-model is as an interval that with a certain level of confidence (95% for this study) covers the true accuracy skill of the model. In this way a confidence interval gives an idea of the stability of a model for a certain measurable statistical value, in this case accuracy.
This study used a confidence interval on accuracy as the deciding factor to determine whether or not a given independent variable's influence had big enough of an effect on the accuracy of a model to be deemed either superior or inferior compared to the performance of the base-model. This was determined by whether or not the confidence intervals of the base-model and the changed variable model overlapped. Only a variable change producing a non-overlapping confidence interval with higher accuracy would be accepted in the final model.

3 Results

3.1 Results From Phase 1: Bootstrap Model Performances

From each performance test of the model in the experiment, 1000 accuracy values were recorded. The measurements for each of these models can be seen in the figure below presented as histograms.



(a) Base acc. (b) No aug. acc. (c) Avg. pool. acc.

(d) Adam acc. (e) Adadel acc. (f) Combi. acc.

Figure 4: Histograms of model accuracies.

3.2 Analysis

The lower and upper boundaries of the 95% confidence interval for each distribution are calculated with the given formula:

$$\overline{p} \pm z_{\alpha/2} \cdot \sqrt{\frac{\overline{p}(1-\overline{p})}{n}} \tag{2}$$

Here, $\overline{p}$ is the sample proportion, $z_{\alpha/2}$ the critical value, and n the sample size.
For the base-model the calculation of lower and upper boundaries of the confidence interval is as follows.

$$0.199 \pm 1.96 \cdot \left(\sqrt{\frac{0.199 \cdot (1 - 0.199)}{1000}} \right) = \begin{cases} 0.174 \\ 0.224 \end{cases} \tag{3}$$

The same estimates were performed for the rest of the models, as seen in the table below.

Tested Model	**Lower Bound**	**Upper Bound**
Base Model	0.174	0.224
No Augmentation	0.179	0.229
AveragePooling2D	0.175	0.225
'Adam' Optimizer	0.170	0.219
'Adadelta' Optimizer	0.175	0.225
Best Model	0.174	0.224

The normal distributions all falls in ranges around 20% accuracy. Since the models work with classification of five possible categories, the accuracy of the models appear to be random. The outcome of the first phase of the experiment therefore leads to the result being inconclusive, whereas no one model can be said to be more fit than another.
The expected outcome written as this study's hypothesis was hereby not found to be correct.

3.3 Results From Phase 3: Final Model Composition

The effect of changing the different independent variables showed to have had no significant effect on the performance of the models. In the light of this uncertainty, where no fitter model could be extracted, phase two was disregarded and the final model was set up using *base*-settings, and trained on a dataset of total 15000 images and validated on 620 images. The outcome of this is displayed below.

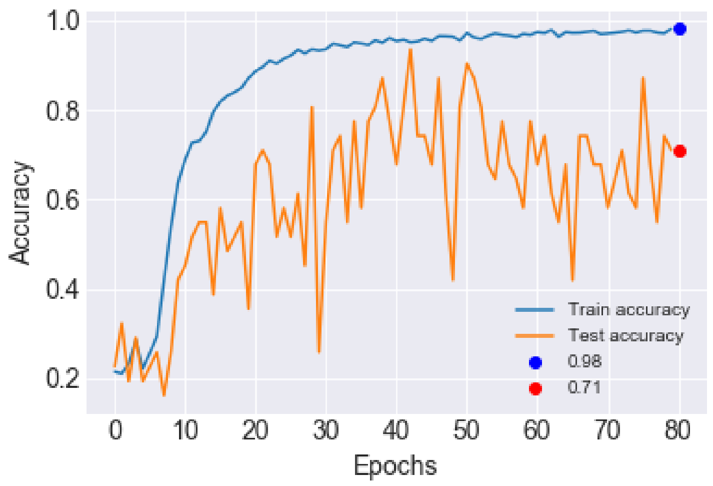


Figure 5: Final model accuracies.

The training-set accuracy shows progress after running 5 epochs and the validation accuracy progresses after running 8 epochs. The training accuracy ends at 98% accuracy and the validation accuracy finishes at 71%.

4 Discussion

Dataset size

The tests of the experiment ran 1000 bootstrap iterations of 150 images per class for each variation of the base-model. One reason as to why these models had similarly random accuracies could be that the training of the models simply did not have enough images from the randomized training-sets to sufficiently adjust the weights of the network. As the weights of the models are randomly initiated for every bootstrap iteration, too few training samples for the weight adjustment results in the weights still being close to random after training and thereby producing random predictions. For future experiments better results would be expected from running a vastly bigger data-set for training, which is also what is seen in the working model which was trained on 15.000 images.

The limitation of the experiments data-set size was caused by a lack of RAM in the hardware, because the program was made to load all images into memory to be able to perform resampling of the data through the scikit learn API[7]. A solution to this problem could be to instead hold an array of integers in memory which could represent each image in the dataset, and then by getting a DirectoryIterator from the Keras API ImageGenerator flow_from_directory function, streaming of the images could be implemented[4].

Number of Epochs per Bootstrap Iteration

The fitting of the models was only done using a single epoch for each bootstrap iteration. As it is seen from the training/test accuracy of the final model in figure 5, the learning starts after 8-10 epochs which makes it plausible that the models tested in the bootstrap needed more epochs to fit to the data.

Final Model

It is reasonable to assume that a dominant factor in the separation between the evaluation accuracies and the performance of the final model is caused by size differences in the two training datasets. However, similarly to the considerations about bootstrapping iterations, the final performance of the model may also be affected by the number of epochs. As seen in figure 5, the validation accuracy peaks at around 40 epochs. Considering this, it may be appropriate to decrease the number of epochs to fully optimize the model for the chosen settings, especially as it is likely that the change in accuracy is caused by the model overfitting to the training data.

Another method of optimizing the model is with Callbacks found in the Keras API. These functions only allow the training to continue if the following iteration is superior to the last one according to the fitting metrics. Callbacks make sure the model does not lower performance after reaching the peak validation accuracy. Had these functions been used during the training of the final model a higher validation accuracy could have been achieved.

References

[1] Agile software development. `https://en.wikipedia.org/wiki/Agile_software_development`, January, 2019. Cited January 23, 2019.

[2] Akash. Kaggle Data. `https://www.kaggle.com/grassknoted/asl-alphabet`, 2018. Cited January 23, 2019.

[3] Jason Brownlee. How to calculate bootstrap confidence intervals for machine learning results in python. `https://machinelearningmastery.com/calculate-bootstrap-confidence-intervals-machine-learning-results-pytho`, June, 2017. Cited January 23, 2019.

[4] Keras API: ImageGenerator. `https://keras.io/preprocessing/image/`.

[5] Jessica Marzen. How we built the asl alphabet game. `https://www.stradigi.ai/blog/how-we-built-asl-alphabet-game/`, December, 2018. Cited January 23, 2019.

[6] Bharath Raj. Data augmentation — how to use deep learning when you have limited data—part 2. `https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced`, April, 2018. Cited January 23, 2019.

[7] Scikit Learn API: resample function. `https://scikit-learn.org/stable/modules/generated/sklearn.utils.resample.html`.

[8] John Elder Robert Nisbet and Gary Miner. *Handbook of Statistical Analysis and Data Mining Applications*. Number ISBN 978-0-12-374765-5. 2009.

5 Appendix

5.1 Project Code

https://gitlab.gbar.dtu.dk/s183921/intelligente-systemer-project

5.2 CSV Data for Accuracy and Loss Training Scores

https://dtudk-my.sharepoint.com/:u:/g/personal/s183915_win_dtu_dk/EfX998QbylFApGdpRuqYThABL_414yG3p3rnaXNNKIZs9w?e=Ch8RrH