Final project DV2599 - Character recognition for the capitalised English alphabet using HOG SVM & CNN

Vincent Honar
Master of Science in Engineering: AI and Machine Learning
Blekinge Technical Institute
Karlskrona, Blekinge
kakh19@student.bth.se

Jonatan Frykmer
Master of Science in Engineering: AI and Machine Learning
Blekinge Technical Institute
Karlskrona, Blekinge
jofy20@student.bth.se

I. Introduction

This report will describe the processes and steps which were taken to compare two Machine Learning models in their ability to correctly classify capital letters from the English alphabet. The Machine Learning models which will be compared are a Convolutional Neural Network (CNN) and a Support Vector Machine using HOG features (SVM). HOG SVMs are known to perform significantly better than traditional svms, comparing these CNNs which are seen as the benchmark for image recognition problems can widen perspectives and increase the reader's perceived tool-set. The data used in this project are from two different datasets, the bulk of the data is from the NIST dataset [1] and data from the EMNIST dataset [2] was used to pad up character with a low occurrence in the first dataset. To compare the models two hypothesis were formulated:

H_0^1: The convolutional neural network and support vector machine will have no statistically significant difference between them in terms of accuracy and f1-score
H_1^1: The convolutional neuralnetwork will have a statistically significant higher performance measure than the support vector machine in accuracy and f1-score.

H_0^2: The support vector machine will either have an equally long or longer training time than the CNN.
H_1^2: The CNN will have a higher training time than SVM which is statistically significant.

Several tests will be performed to test these hypothesises, the Shapiro-Wilk's test and Levene's test will be conducted to see if our data satisfy the ANOVA assumptions [3]. If both tests are passed then the parametric One-tailed Independent samples t-test will be used to see if there are any statistically significant differences in the metrics. In case one of the aforementioned tests do to not pass the non-parametric Mann-Whitney U test will be performed. If the selected test indicate statistical significant significant difference for either performance metrics (H_0^1) or training time (H_0^2) the respective null hypothesis will be rejected. There are multiple benefits and use cases for the field we are researching, one notable mention is the ability to quickly digitalize physical text documents.

II. Background

A. Stratified k-folds cross-validation test

Stratified k-folds cross validation divides a data-set containing classified samples in 10 roughly equally sized sets. One fold is selected as the training fold whilst the other is used for testing. The folds retains the same ratio between target classes as the data set it was derived from.

B. HOG Support Vector Machine

A Support Vector Machine is a supervised machine learning model which classifies data by plotting all data-points in an n-dimensional space. The axis in each dimensional can be understood as a feature of a data-point. To distinguish between classes multiple hyperbolas are drawn separating all data-points within a certain class from the different classes. In this report a HOG SVM is used where HOG stands for Histogram of Oriented Gradients. Which is a technique that tries to represent the shape of an object within an image by creating gradient orientations which try to match the intensity of the direction of the object.

C. Convolutional Neural Network

A Convolutional Neural Network is a type of Neural Network which is especially good at image recognition. It differs from a conventional Neural Network mainly by how the data is processed by the network. Another difference that a CNN has convolutional which does feature extraction automatically unlike a support vector machine.

D. Testing hypothesis

To test the hypothesises of the report an either parametric or non-parametric test will be used to see if there are any statistically significant differences in the various performance metrics generated by the Stratified k-folds cross-validation test.

E. Testing for the ANOVA assumptions

In order to use a parametric test the ANOVA assumptions must be true. The satisfy the ANOVA assumptions the following statements must be true, the data must be independent, normally distributed and homoscedastic [3].

F. Mann-Whitney U test

The Mann-Whitney U test is a non-parametric test which aims to determine if two independent samples are from the same population or not. It does it by first combining and ranking each value from both samples then it takes the sum of the ranks from each sample and compares it with the other sum from the other sample.

G. One-tailed Independent-samples t-test

A One-tailed Independent-samples t-test is a parametric test which is used to determine if there exists a statistically significant difference between the means of two independent sample groups. The test uses a t-value which is calculated by subtracting the mean of one group with the mean of the other group. If this t-value is other a certain critical threshold there exist a statistically significant difference between the groups. The threshold can be found by looking up the Student's t-table. This test works best when the sample size is fairly small which is true in our case.

III. Methodology

A. Exploration

The A-Z characters data-set's structure was examined to understand how to handle the data. Properties such as how pixel-positions are represented and the label column were identified and noted. A bar graph which described the class distribution was created. Twenty-seven randomly selected letters were visualised to examine for noise, mislabelling, and or if less common writing-styles such as cursive letters existed. Letters which either had a resemblance or possessed similar features to more populous letters were presented together in search of things to look for under evaluation (e.g., if looking at the model's ability to distinguish between L:s and I:s should be a thing of concern). A check for duplicates was done along with presenting the amount of duplicates for the entire set as well for each label. The class label with the lowest amount of duplicates had all of its duplicate letters presented to guarantee that the the instances were indeed marked correctly.

B. Data cleaning

Upon investigating a subset of all duplicates, they were dropped from the main data-set and a new plot describing the distribution of samples among class-labels was created to repeat the previously aforementioned goal. Neither any significant noise was found nor mislabelling was found that needed correction. Missing values were not inspected as the website providing the set had a preview which stated that no missing values were present.

C. Integration with EMNIST-byclass

Classes which were under-represented in the set received more samples from the other NIST-derived set to reduce class imbalance. An arbitrary threshold of less than 10 000 samples was used to determine if a label could be seen as under represented.By using the aforementioned steps under exploration to understand the data, the EMNIST-byclass data-set was found to have three issues which hindered integration without processing the data first.

Both digits and lowercase letters were present, this shifted all uppercase letters to another class-label value which does not coincide with A-Z. By extracting all uppercase letters into a separate data-set and shifting the values to match A-Z this issue was resolved.

The columns were renamed in accordance with the naming convention present in the destination data-set using a mapping of A-Z's column names to the corresponding EMNIST-column.

Uppercase letters had their pixel positions flipped on the x axis and rotated 90 degrees. Reshaping the images into 2D matrices and Transposing said matrices to then return them to their one-dimensional form solved this issue.

Since both sets were derived from The NIST-19-special-database the risk of introducing duplicates was high. Duplicates were removed again and a brief repeat of the first steps in the exploration section were conducted. With the final size of the set finalised, a plot was created to see if enough of an imbalance existed to warranted using class weights based on samples. Weights were thus created.

D. Preprocessing of images for models

The models, whilst using the same data, require the input to be in the form of 2-dimensional matrices. Samples are thus reshaped into a 28x28 matrix before being fed into the models. SVMs do not conduct feature extraction by themselves. To create a HOG SVM, one needs to process the images themselves.

1) Extracting HOG features: Being unfamiliar with HOGs in general, an exploratory approach was taken in finding the correct settings. A particular sample,a letter which has a mix of sharp angles and soft curves, was selected and presented under 72 different configurations for HOG extraction at a time. This was done repeatedly and with different settings. Once a setting was found to be adequate, the configuration was applied to a random selection letters to see if the settings resembled the example given by Galkissa [**?**] for all letters. A second opinion was gathered from a senior lecturer at Blekinge Technical Institute, Hüseyin Kusetogollari, to confirm that these settings sufficed

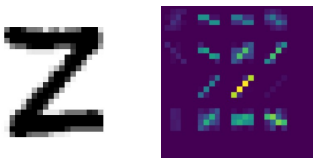

Fig. 1. An example of how the image is transformed when extracting HOG features

E. Model configuration

1) CNN: The CNN follows the same node configuration as presented on tensor-flow's example in the documentation for such a model [6] with three minor changes, namely the input size, amount of nodes on the output layer and its activation function. Since the data set consists of data 28x28 images the input layer was modified to allow for such input. 26 classes/letters are supposed to be the output of the result, thus 26 nodes are used in the output layer to generate a classification. These nodes use the Softmax activation function to return a list of probabilities for each label. No further changes were made as any other changes could not be motivated.

The model was trained with weights and a batch-size of 64. Whilst a lower batch-size does increase performance, the risk of it introducing noise is also higher. With a dataset of 200,000 samples, there was no feasible way to thoroughly check for noise, thus a batch-size of 64 reduces the chance of noise causing issues whilst remaining low enough to give the model enough batches to adjust with.

2) SVM: The SVM was weighted with the same sample based weights as the CNN. HOG SVM was decided to create a more interesting comparison as an SVM alone does not do feature extraction.

F. Training and testing models

Five dictionaries with the same structure, the model name being the key which leads to an empty list, were created for the purpose of storing various metrics and outputs from the models. Stratified 10-fold-cross-validation was used to gather enough sample points from each sample to use in testing. The following steps were repeated for each fold:

1) Get testing and training folds
2) Reshape data into 28x28 matrix
3) Create a copy of the data and extract HOG features
4) Create new instance of a SVM
5) Train SVM with HOG data and measure training time
6) Test SVM with HOG data used for testing
7) Do a classification report for class-wise measures, measure accuracy & f1-score and store it in a dictionary.
8) Create a new CNN instance
9) Divide training fold into training and validation data (85/15 split)
10) Train The CNN and measure its training time
11) Test CNN on testing data from the test fold
12) Do a classification report for class-wise measures, measure accuracy & f1-score and store it in a dictionary.
13) Store the resulting predictions and models in a dictionary.

G. Statistical tests

Levene's and Shapiro-Wilk's test were conducted on each of the three metrics. Three one-tailed Independent-samples t-tests were done to either sustain or reject the null hypothesis posited.

H. Preprocessing of images in the demo application

An app was made to test the models in a real environment. The application allows a user to draw a combination of letters and have the model guess what was written upon pressing the recognise button. This is achieved by cropping each letter into the smallest possible image which contains the letter, pad the sides to adhere to a square shape and downsize the image to 28x28. Each letter in the drawing-board receives the previously mentioned treatment and is then sent to the model to either be recognised or have its HOG features extracted and then recognised one by one.

IV. Results & Analysis

All metrics passed both Levene's and the Shapiro-Wilk's test which meant that ANOVA assumptions had been reached. The average execution time for CNN and SVM respecitively was roughly 19.5 minutes and 11.7 minutes.

TABLE I
TIME

Fold	CNN	SVM
0	1154.81	729.99
1	1151.43	695.30
2	1141.61	681.40
3	1142.89	679.15
4	1146.86	687.63
5	1158.63	701.71
6	1151.15	711.01
7	1144.15	698.83
8	1219.25	710.26
9	1264.75	712.59
avg	1173.87	700.07
std	41,02	15,77

The p-value yielded from the Independent-sample t-test is $1.08 \bullet 10^{-17}$ which is far below the requirements to reject the null hypothesis. This difference could stem from the fact that Sklearn, the library used for the support vector machine, and Tensorflow have different hardware support. But this fact should be in the neural network's favour as it can utilise both the GPU and CPU whilst the SVM is limited to the CPU [7]. A likely explanation to the discrepancy is the number of epochs conducted by the CNN. 15 epochs is the equivalent to passing an entire fold through the model 15 times, whilst this is necessary for the CNN to achieve a high accuracy it impacts training time considerably. An arbitrarily lower amount of epochs could be assigned to the model for training at the cost of performance. Nonetheless 15 epochs could either be too high or a good fit as the CNN performed similar to the support vector machine at all other metrics. This results in a possibly good comparison for how much computing power is required to yield similar results. Testing at with different epochs would be required to conclusively get an answer.

Neither fscore nor accuracy yielded passed the Independent-sample t-test with a p-value of. Both the convolutional neural network and support vector machine with HOG features performed at a high accuracy. The sample size of each fold could

possibly favour the support vector machine [8] as they tend to perform better than CNNs with lower samples. Considering the marginal difference in performance, this factor should be considered.

Whilst the tests may not have passed the results yielded by the folds are good results. Thus some scepticism should be applied to the percentages as they could be skewed by the over-represented classes although the weighted f1score contradicts this notion as it is roughly 0.2% higher than the unweighted value for both models.

TABLE II
F-SCORE

Fold	CNN	SVM
0	0.98334	0.98106
1	0.97981	0.98151
2	0.97948	0.97764
3	0.98028	0.97870
4	0.97554	0.97818
5	0.97583	0.98047
6	0.97811	0.97872
7	0.98087	0.98084
8	0.97929	0.97865
9	0.97688	0.97605
avg	0.97909	0.97838
std	0,00241	0.00174

Two confusion matrices were created in the accompanied notebook which examined the models performance on the first fold class-wise. The support vector machine as well as the convolutional neural network had the highest performance on the letter T. Where the models performed poorly differed though as not only were O:s and Q:s were mistaken by the SVM at a higher rate but 1.5% of all E's were misidentified as F. The convolutional neural network struggled with differentiating Y:s and V:s as well as A:s and H:s which do have a resemblance to eachother. Only 96.3% of all G:s were correctly identified by the SVM, which was by far its worst performing letter and the CNN's worst performing letter in the first fold was R at 96.6%. R was misidentified especially as K and A. These trends could be specific for the fold but have reasonable explanations.

TABLE III
ACCURACY

Fold	CNN	SVM
0	0.98336	0.98108
1	0.97986	0.98152
2	0.97955	0.97769
3	0.98034	0.97874
4	0.97558	0.97820
5	0.97589	0.98047
6	0.97813	0.97874
7	0.98088	0.98084
8	0.97928	0.97864
9	0.97687	0.97605
avg	0.97909	0.97838
std	0,00241	0.00175

The application which was made as a way to try the models in a real-world scenario yielded similar results regardless of model. The SVM had an easier time recognising I:s than the CNN. The application consistently identifies the correct letters. In the case where the an incorrect letter is identified, the correct tends to be among the other 2 predictions of the model. See figure 2 for an example.

V. CONCLUSION

H_0^1 could not be rejected, even when conducting a two-tailed t-test the result p-value was above 0.05 indicating that the models did not have statistically significant differences in performance. It cannot be ascertained whether this lack of a difference exists because the models are equally good or whether the task or hand favoured one algorithm over the other [8].

H_0^2 was soundly rejected as SVM trained on average 470 seconds faster than CNN. This was to be expected as CNN's have a higher complexity than support vector machines Nontheless both CNN and SVM perfomed at a high accuracy rate with this data-set. In this particular scenario, the HOG SVM kept up with the CNN and even outdid it by a small sliver in terms of performance. Further experimentation using larger sample sizes would be required to draw more meaningful conclusions.

It is needless to say but character recognition has many use areas and can be applied in both novel, such as the drawing application, and serious matters for example transcribing documents.

VI. CONTRIBUTION

Most work was done through pair-programming where one person, usually Honar, wrote the code whilst the other, Frykmer, proofread it and looked for more information about the task at hand. This dynamic disappeared during the later stages of finishing the application as Frykmer took initiative to implement several features such as swapping models and a result history. Tasks were instead delegated to each member to more effectively utilise time.

A. Exploration

The initial exploration was done together as both the participants needed a greater understanding of the data which was being handled. After understanding the structure of the set and creating the bar graph of the label distribution, Honar created the functions to view samples as preview images.

B. Data integration & Cleaning

The integration of the two sets was done by Honar as Frykmer worked with finalising the application. Data cleaning was also done by Honar with input from Frykmer as this issue was found rather late in the project's life cycle.

C. Application

Since neither member had ever worked with tk-inter, both members partook in creating the demo. The processing of the letters were done pair-programming whilst the output of the results in the window along with the ability to classify a presaved file was done by Honar. Switching models, clearing outputs, getting prediction history and making the program work with a SVM model was done by Frykmer alone.

D. Code implementation

Creating the hog features and experimentation to find the right settings was done together as neither member had prior experience with this feature-extraction technique. The confusion matrices present in the notebook were created together, this can be said by everything unless specifically stated otherwise. The training and testing with folds along with setting up the statistical tests was done by Honar whilst Frykmer commented the application.

E. Evaluation

Both members looked for testing metrics to use for the evaluation and discussed which statistical tests to do. Honar proposed first testing for the ANOVA assumptions and either doing a indepedent t-test or Mann-Whitney U test depending on the results.

F. Report writing

During the report writing Frykmer wrote the Introduction as well as Background while Honar wrote about the Methodology, Evaluation and Results & Analysis. Both members wrote this section of the report together and proofread the other member's sections while giving possible revisions or improvements to the text. Frykmer created the tables and images present in the report while the data needed to do so was formatted and provided by Honar. Most of the references were handled by Frykmer with input from Honar.

REFERENCES

[1] C. Crawford. EMNIST (Extended MNIST), Version 3. Retrieved 10-Jan-2023 from https://www.kaggle.com/datasets/crawford/emnist?resource=download&select=emnist-balanced-test.csv.

[2] S. Patel. A-Z Handwritten Alphabets in .csv format, Version 5. Retrieved 18-Nov-2022 from https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format.

[3] D. Navarro. (03-02-2022). Intro statistical analysis [Slideshow]. Blekinge Technical Institute. https://bth.instructure.com/

[4] M. Galkissa. Training SVM classifier with HOG features, Version 2. [Accessed: 12-Dec-2022] on https://www.kaggle.com/code/manikg/training-svm-classifier-with-hog-features

[5] H. Kusetogullari. Personal Communication, 04-01-2023.

[6] Tensorflow CNN Tutorial, Convolutional Neural Networks, TensorFlow, https://www.tensorflow.org/tutorials/images/cnn, 2021.

[7] Scikit Learn frequently asked questions: will you add gpu support, Scikit-learn, https://scikit-learn.org/stable/faq.html#will-you-add-gpu-support, 2021.

[8] Pin Wang, En Fan, Peng Wang, Comparative analysis of image classification algorithms based on traditional machine learning and deep learning, Pattern Recognition Letters, Volume 141, 2021,Pages 61-67, ISSN 0167-8655, https://doi.org/10.101/j.patrec.2020.07.042. (https://www.sciencedirect.com/science/article/pii/S0167865520302981)

Fig. 2. Example output of the APP with CNN