Battle of the Giants

CNN vs. SVM



Isabella Frid

EC Utbildning

Machine Learning

202403

# Abstract

In this project I developed a knowledge I didn’t intended to learn, deep learning, and more specific – a CNN model. The Convolutional Neural Network (CNN) achieved a test accuracy of around 99% and proves its effectiveness in image recognition tasks. To get this result in comparison I used the SVM classifier (Support Vector Machine) which received an accuracy of 97%. Even with its lower score, I still believe that the SVM did a fair fight and got itself a win.

# Abbreviations

CNN = Convolutional neural network

DL = Deep learning

ML = Machine learning

SVM = Support vector machine

Table of Contents

[Abstract 2](#_Toc162129470)

[Abbreviations 3](#_Toc162129471)

[1 Introduction 1](#_Toc162129472)

[1.1 Overview of Convolutional Neural Networks and Support Vector Machine 2](#_Toc162129473)

[2 Theory 3](#_Toc162129474)

[2.1 Evaluation 4](#_Toc162129475)

[3 Method 5](#_Toc162129476)

[4 Results 7](#_Toc162129477)

[5 Verdict 8](#_Toc162129478)

[6 Questions 9](#_Toc162129479)

[7 Self-Evaluation 11](#_Toc162129480)

[Source list 12](#_Toc162129481)

# Introduction

Machine learning and Deep learning are both parts of artificial intelligence, and enables computers to learn from data and make predictions or decisions without being explicitly programmed. One area that has been increasingly popular the last couple of years is image recognition and classification. CNNs are a class of deep learning models designed to analyze visual data, while SVMs are more versatile and known to be less effected by noise and less prone to overfitting as CNNs are.

Understanding and implementing CNNs and SVMs are crucial in today’s technological revolution due to the increasing demand for imaging processing application. The future with self-driving cars and where AI systems can analyze medical images with precision is already here, making CNNs and SVMs relevant areas of studies in various industries.

The purpose of this report is to explore and compare the implementation of CNNs and SVMs for image classification tasks using the MNIST dataset, consisting of 70 000 handwritten numbers.

1. How does the performance of a CNN model compare to a traditional machine learning model as Support Vector Machines (SVMs) on the MNIST dataset?
2. What are the advantages and limitations of using CNNs for image classification tasks compared to other machine learning techniques?

## Overview of Convolutional Neural Networks and Support Vector Machine

Convolutional Neural Networks are subset of machine learning and uses deep learning algorithms. CNNs consist of different layers and uses three-dimensional data for image classification and object recognition tasks.

Key characteristics:

1. **Convolutional layers:** CNNs uses convolutional layers to apply filters to input of images to extract features in a hierarchy.
2. **Pooling layers:** Pooling layers works similar to convolutional layers, but reduces the dimensions and can help reduce overfitting and complexity.
3. **Fully connected layers:** These layers are helpful in the final stage of a CNN to perform classification based on extracted features.

Support Vector Machines are a class of supervised learning algorithms used for classification and regression tasks. SVMs are effective in high dimensional spaces and can handle both linear and non-linear classification tasks.

Key characteristics:

1. **Hyperplane:** SVMs works by finding the hyperplane that best separates the classes in the feature space.
2. **Kernel trick**: By using kernel trick SVMs can handle non-linear decision by mapping the input data into higher dimensional spaces.
3. **Margin maximization**: SVMs aim to maximize the margin between classes, and works great against overfitting.

# Theory

CNNs are a type of neural networks and are known for their complexity and ability to learn patterns, but a question arises – is more complex always better? Increasing the complexity of a model may improve its ability to capture difficult features, however it may also come with drawbacks. I’m talking about longer training time and a higher risk of overfitting, which were something I had to deal with. On the other hand, simpler models like SVMs have fewer parameters and are – by experience – easier to interpret and train.

In this report I will explore the trade-off between model complexity and performance by comparing the result of CNNs and SVMs.

## Evaluation

Validation is a crucial step in machine learning workflow and the primary goal is to evaluate how well the model generalize new and unseen data. By testing a model on a separate dataset that it has not been trained on, we can gain insights into its ability to make accurate prediction. Validation also helps identify potential issues such as over- and underfitting.

Because I’m using a model from DL and another from ML I got some problem with finding a good way to validate the results using the same method. I wanted to use K-fold cross-validation as it seem to be a good fit for both models, but for some reason I couldn’t make the code work for CNN.

Instead, I will be using model evaluation to determine how effective both models are. I will also include classification reports, consisting of F1-score, Recall, Precision and Support.

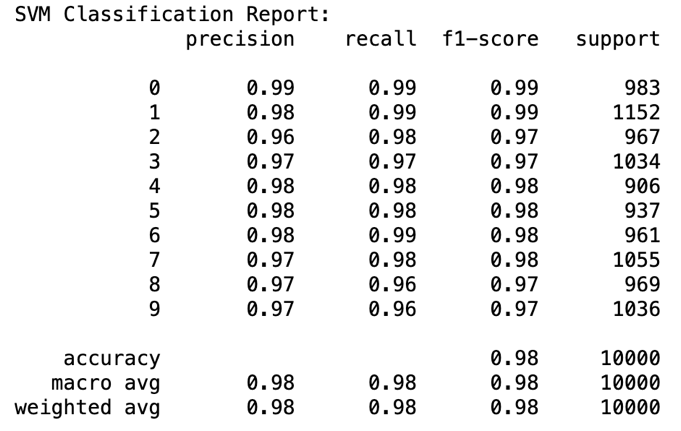
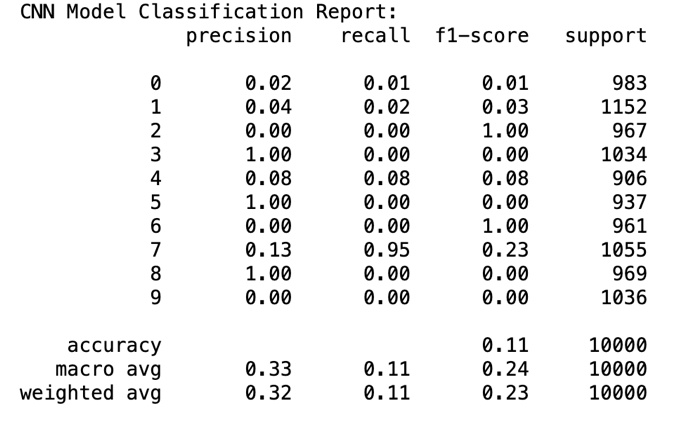


Figure 1 - To the left, CNN Classification report

Figure 2 - To the right, SVM Classification report

# Method

For this tasks we were told to use the MNIST dataset which includes images of 70 000 handwritten number, and my first instinct was to Google “Which model is best for image recognition” which gave me the result of CNN and SVM. Here is where I first go wrong, and I don’t start to read about the chosen models, but instead started to look up code examples. Had I open some of the links with my Google search I probably would have seen that CNNs are part of deep learning and not machine learning. What I did instead was to search for “CNN” in the book “Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow and found myself satisfied when I saw 139 results and started skimming through the pages.

In Jupyter I began with loading the dataset and sort the data and targets into X and Y, and then into train and test sets, where 10 000 images where set aside for the testing.

I felt confident and started with the CNN and created a function with different layers and began training. The result was bad. The score from training were up in the nineties but the validation were in the 10% - a typical case of overfitting. So I began to add layers, dropouts, early stopper, Learning rate scheduler, Adam optimizer and last Image data generator. The result got even worse. Both training and test were set to 11% at the highest. No matter what changes I did make any big difference on the numbers and I started to rethink my choice of studies.

It got worse. I thought that something might be wrong and I reran all cells and kernels and all the code that worked perfectly – with bad result, but still worked – stopped working. Error code after error code. Error codes that didn’t even make any sense. The function had apparently, somehow, stopped existing. As a former trained tech support I lastly restarted Anaconda. Jupyter. And my Macbook. It did nothing.

My soul started to crumble. I tried to save a copy of the code, rewrite the code, search through Google as my life was pending on it, and I stumble upon endless of forums with the same problem and no one seem to have found a reason or a solution.

I opened an empty document and I started to write my code from the beginning. I made some changes, and tried my function with just the layering and dropout, and I started to see the light through it all. 98.7%. I added the fixes I had earlier wrote to fix the overfitting - that now didn’t exist - and the results lands on 99.15%.

When my heartrate was back to normal I felt confident enough to give the SVM model a try. I kept it simple, just train and validation, and the results shows 97.73%. Or 98% if I use recall or F1 score.

That was good enough for me.

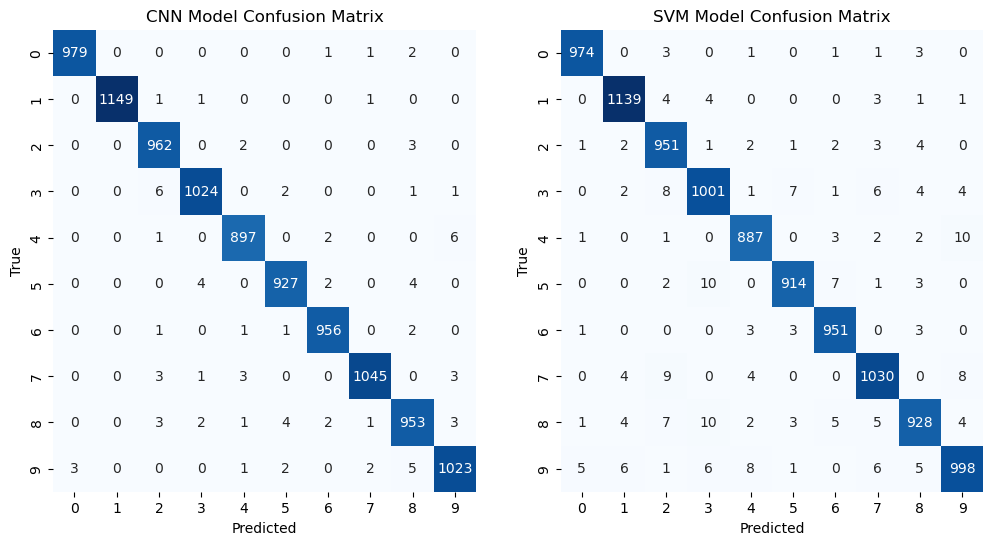


Figure 3 - Confusion Matrix of CNN and SVM results

# Results

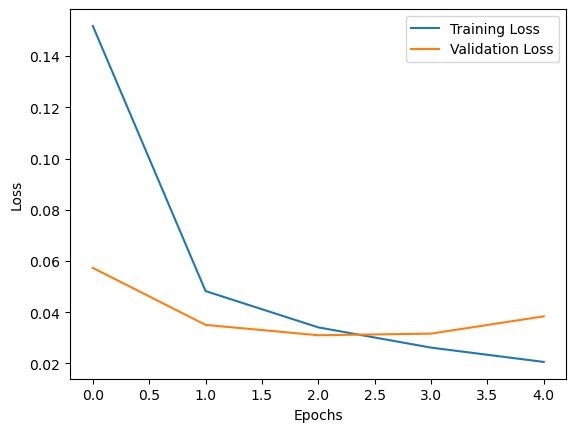
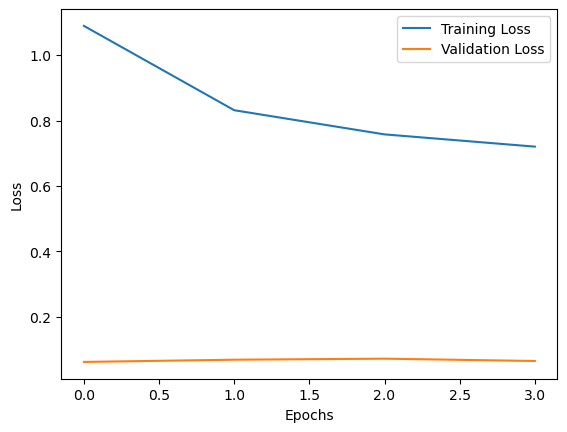
 

Figure 4 - To the left, first attempt with CNN

Figure 5 - To the right, the second attempt with CNN

When we revisit the result we can easily see that the CNN model worked perfectly, and even found room for improvements. The first result landed on 98.7% accuracy and with some added complexity it rose up to 99.15%. But I wanted to put it to a test. I choose a random image and let the model predict. The image showed us a 3 but the prediction was set to 9. Does the CNN model not work?

I take responsibility for the test. The code looks good to me, but no matter how many time I rerun the cell or change the image, the prediction remains the same. I will keep this part out of my verdict.

The SVM classifier was quick and easy, and return an accuracy of 97.73%. I chose not to make any changes for improvement to test my thesis.

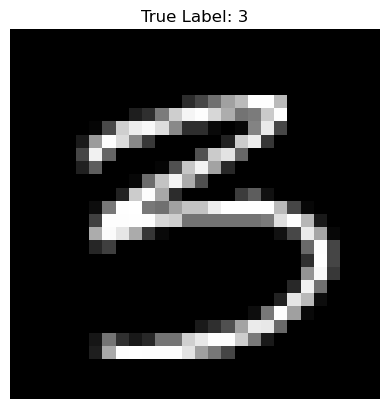


Figure 6 - Image 100 presenting the number 3

# Verdict

So, how did the performance of a CNN model compare to a traditional ML model as SVM?

Both models performed great and I did think to see more of a difference between the results. In numbers the CNN is only 1.42 percentage points better. When looking at the results of the classification report we can clearly see that the CNN model doesn’t do well at all, but I’m sensing that I might got myself in the same problem as I had during my coding. At this point I want to save what’s left of my mental health and don’t want to risk all code by rerunning the code and kernels and will therefore leave this part out. But I haven’t really forgotten about the bad prediction CNN made and since the SVM classifier actually predicted the image 100 correctly, I would say SVM won this round.

What can we say about the advantages and limitations of using a CNN model?

We can say a lot of great things about the usage of a CNN model, from being able to learn complex patterns with spatial hierarchies to scale invariance, allowing the model to learn features at multiply scales. The downside with the model is that its prone to overfitting and usually requires large amounts of labeled data for training – which also can be time consuming.

With this type of database I would prefer to use SVM for all the reasons I previous stated throughout the report. It’s easy, quick, beginner friendly and in my case, less problematic. I might be bias – and a bit traumatized – but with such a small difference in the results and the fact that the SVM could predict the image when CNN couldn’t, I’ll be sure to go to with a SVM in the future.

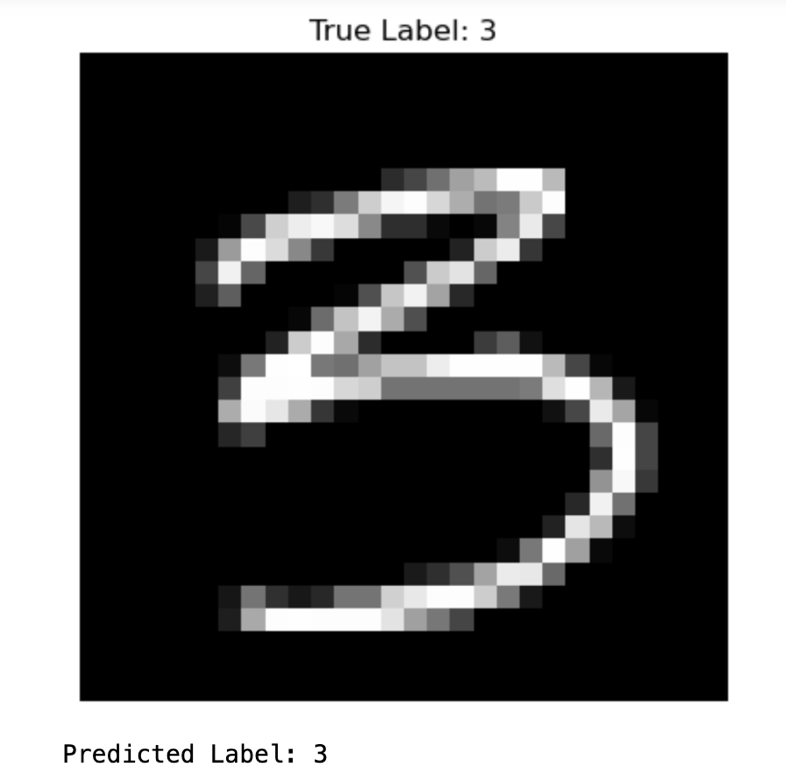


Figure 7 - SVM prediction on image presenting number 3

# Answers to Questions

1. Kalle delar upp sin data i ”Träning”, ”Validering” och ”Test”, vad används respektive del för?

* Träningsdata används för att träna modellen, och efter varje träningsomgång kan modellen utvärderas på valideringsdatan för att bedöma generaliseringsförmågan, och testdata används sedan som ny och osedd data för att utvärdera modellen och dess prestanda.

1. Julia delar upp sin data i träning och test. På träningsdatan så tränar hon tre modeller; ”Linjär Regression”, ”Lasso regression” och en ”Random Forest modell”. Hur skall hon välja vilken av de tre modellerna hon skall fortsätta använda när hon inte skapat ett explicit ”validerings- dataset”?

* Julia kan använda till exempel Cross-Validation för att utvärdera modellernas träningsdata. Ett annat exempel är RMSE som utvärderar testdatan.

1. Vad är ”regressionsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden?

* Regressionsproblem är en typ av problem där målet är att förutsäga en kontinuerlig numerisk variabel. För dessa problem kan man bland annat använda Linjär regression eller Lasso. Ett område där vi kan använda dessa modeller är när vi vill kunna förutse bostadspriser eller aktiekurser.

1. Hur kan du tolka RMSE och vad används det till?

* RMSE står för Root Mean Square Error och kan användas för att mäta en modells prestanda. Den mäter genomsnittlig kvadratisk skillnad mellan de faktiska och förutsagda värdena från modellen. Ett lägre värde kan innebära att modellen har mindre fel i förhållande till de faktiska värdena, medans ett högt värde istället kan ha större fel och är mindre noggrann.

1. Vad är ”klassificieringsproblem? Kan du ge några exempel på modeller som används och potentiella tillämpningsområden? Vad är en ”Confusion Matrix”?

* Klassificieringsproblem är en typ av problem där man vill kunna förutse en kategori eller klass, till exempel True/False. Logistisk regression, Support vector machine och K-Nearest Neighbors är modeller man kan använda för dessa typ av problem, och kan användas för tex. Spamfiltrering av e-mail eller för medicinsk diagnostik, där patienten antingen är sjuk eller frisk. Confusion Matrix är en tabell för att mäta en modells prestanda och visar antalet korrekta och inkorrekta klassificeringar modellen gör (True positive, True negative, False Positive och False negative).

1. Vad är K-means modellen för något? Ge ett exempel på vad det kan tillämpas på.

* K-means är en algoritm för klusteranalys och grupperar in observationer i olika kluster baserat på sina likheter. K-means kan användas för marknadssegmentering för att se kunders köpvanor baserat på vilken stad kunden bor i.

1. Förklara (gärna med ett exempel): Ordinal encoding, one-hot encoding, dummy variable encoding. Se mappen ”l8” på GitHub om du behöver repetition.

* Ordinal encoding är en teknik för att dela in unika kategorivärden (när det finns en rangordning mellan kategorierna) i heltal, tex. Klädstorlekar som S, M och L kan delas in som 1, 2 och 3.
* One-hot encoding är en teknik som skapar en binär variabel för varje unikt kategorivärde när det inte finns en rangordning mellan kategorierna. För varje observation är det endast en av dessa variabel som får värde 1, och resterande får värde 0.
* Dummy variable encoding är en typ av One-hot encoding men där en av variablerna tas bort.

1. Göran påstår att datan antingen är ”ordinal” eller ”nominal”. Julia säger att detta måste tolkas. Hon ger ett exempel med att färger såsom {röd, grön, blå} generellt sett inte har någon inbördes ordning (nominal) men om du har en röd skjorta så är du vackrast på festen (ordinal) – vem har rätt?

* Nominal data är kategorier utan rangordning, medan ordinal data har en rangordning. Färger i sig har ingen rangordning, alltså nominal, men eftersom Julia anser att färgen röd skulle vara finare än färgerna grön och blå så rangordnas färgerna efter Julias preferenser. Detta innebär att Julia anser att färger skulle vara ordinal. Jag håller därför med Göran, men är väl egentligen en tolkningsfråga.

1. Vad är Streamlit för något och vad kan det användas till?

* Streamlit är ett open-source python bibliotek och kan användas för att skapa applikationer för maskininlärning och för dataanalyser.

# Self-Evaluation

I’ve come across a couple of problems during the work with my code as I stated under “Method” in this report. When the code stopped working the first time I resolved it by starting all over again. Is it a great solution? Probably not, but with my bad time management I’ve had for this task, I have only myself to blame. I think the code stopped working again, but I feel like I’ve done the best I can and will leave it as is.

I know I should’ve chosen two models from machine learning, but I hope I’ve proven my skills with this task together with the previous one using linear regression and lasso. I think – and hope – I’ve shown skills and knowledge enough for passing this course.

The report is kept short and concise as requested, thank you for reading.

# Source list

IBM (year unknown). Convolutional neural networks. <https://www.ibm.com/topics/convolutional-neural-networks>. (Retrieved 2024-03-22)

IBM (year unknown). Support vector machine. <https://www.ibm.com/topics/support-vector-machine>. (Retrieved 2024-03-22)

Datagen (year unknown). Convolutional neural networks. <https://datagen.tech/guides/computer-vision/cnn-convolutional-neural-network/#> (Retrieved 2024-03-22)

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2nd Edition. O’reilly Media Inc.