Kursprojekt i Maskininlärning

Bildklassificering på MNIST-datasetet med maskinin- och djupinlärningstekniker

Maria Lagerholm
Data Science-programmet, EC Utbildning

Mars 2025
Göteborg, Sverige

Abstract

This project introduces a Handwritten Arithmetic Solver that can recognize digits and basic math symbols to solve simple equations. Users draw directly on the screen. The app segments each character, predicts their values and calculates the result. I tested three machine learning models - Logistic Regression, SVM and CNN. CNN outperformed the others with the highest accuracy. To improve performance, I extended the MNIST dataset by adding synthetic plus and minus signs, and applied data augmentation like rotation and scaling. The application was built with Streamlit and runs both locally and on the Streamlit Cloud. Overall, this project shows how model choice, creative data preparation, and support from generative AI tools and my teacher, Antonio Prgomet, can lead to a practical solution.

Förkortningar

OHR – Online Handwriting Recognition
AI – Artificiell Intelligens
AGI – Artificiell General Intelligens
ANI – Artificial Narrow Intelligence
GPT – Generative Pre-trained Transformer
CNN – Convolutional Neural Network
SVM – Support Vector Machine
OCR – Optical Character Recognition
ANN – Artificiella Neurala Nätverk
ViT – Vision Transformer
PCA – Principal Component Analysis
RBF – Radial Basis Function
GUI – Graphical User Interface
API – Application Programming Interface
MNIST – Modified National Institute of Standards and Technology
ReLU – Rectified Linear Unit
GUI – Graphical User Interface
RNN – Recurrent Neural Network (nämns inte direkt men kan vara relevant)
GAN – Generative Adversarial Network
VAE – Variational Autoencoder
LSTM – Long Short-Term Memory
BERT – Bidirectional Encoder Representations from Transformers

Innehållsförteckning

Inledning

Detta projekt fokuserar på Online Handwriting Recognition (OHR) med tillämpning av maskininlärning och djupinlärning, med inriktning på handskriftsigenkänning.

Syftet med projektet är att skapa en webbaserad applikation där användaren kan skriva enkla räkneuppgifter för hand på skärmen. Applikationen ska känna igen siffror samt plus- och minustecken, räkna ut svaret och visa det på skärmen.

För att uppnå detta mål ska jag följa dessa steg:

1. En litteratursökning som omfattar vetenskapliga artiklar, kodexempel, handledningar samt tillämpning av generativa AI-modeller för att identifiera optimala lösningar.
2. Undersöka vilken typ av modell som presterar bäst för att känna igen handskrivna siffror.
3. Välja en modell som har över 98 accuracy på testdata.
4. Använda Scikit-learn och TensorFlow för maskininlärning respektive djupinlärning. Använda Grid Search för att hitta de bästa hyperparametrarna.
5. Förbättra datasetet genom att lägga till variationer (data augmentation) för att förbättra modellens prestanda på att känna igen verkliga tecken.
6. Lägga till syntetiska plus- och minustecken i datasetet (I MNIST format) för att modellen ska kunna lösa räkneuppgifter.
7. Skapa en repo på GitHub och ladda upp en detaljerad, väll struktureard projekt där man kan lätt reproducera min experiment.
8. Bygga en webbaserad applikation med Streamlit där användaren kan skriva siffror för hand, få dem tolkade och se resultatet.
9. Testa och förbättra applikationen så att den fungerar bra för alla användare inklusive mobilanvändare.

Projektet har även som mål att ge mig praktisk erfarenhet av att tillämpa olika maskininlärnings- och djupinlärningsmodeller i verkliga scenarier.

1 Teori

1.1 Machine learning, Deep Learning, AI

Människans hjärna är mycket bra på att känna igen mönster, lösa problem, anpassa sig till nya situationer och lära sig av erfarenheter. Machine learning uppfanns som ett resultat av försök att efterlikna mänskligt lärande och automatisera mönsterigenkänning genom algoritmer. År 1959 myntade Arthur Samuel, en IBM-anställd och pionjär inom datorspel och artificiell intelligens (AI), termen ”machine learning” och utvecklade ett program som kunde spela dam, vilket demonstrerade hur maskiner kunde lära sig genom erfarenhet (Introduction to Machine Learning, 2003). En machine learning-modell består av matematiska formler och parametrar som beskriver relationer i data för att möjliggöra prediktioner, klassificeringar, mönsterigenkänning och informerat beslutsfattande.

Deep learning är en gren av machine learning som använder djupa neurala nätverk för att identifiera komplexa mönster i data. Ordet "djup" i deep learning härstammar från användningen av flera lager i neurala nätverk, vilket möjliggör djupare representationer av data. Eftersom deep learning använder djupa neurala nätverk som kan identifiera komplexa mönster och samband utan att behöva manuellt konstruerade funktioner. Klassiska maskininlärningsmodeller kräver manuellt skapade funktioner för att extrahera relevanta egenskaper, medan deep learning självständigt lär sig dessa direkt från data (*Djupinlärning jämfört med maskininlärning - Azure Machine Learning*, 2024).

Både DL och ML är inom området artificiell intelligens (AI), som också innefattar regelbaserade system, kunskapsrepresentation och evolutionära algoritmer. Mycket forskning görs för att utveckla AI till AGI (Artificiell General Intelligens), men än så länge saknar AI förmågan till generell förståelse och självständig problemlösning som människor har. Idag tränar man en AI-modell för en specifik uppgift, och detta kallas för smal AI (Artificial Narrow Intelligence, ANI) (Wheeler, 2025).

Många trodde att för att skapa generativ AI, den som kan producera text, bilder eller ljud, behövde man förstå kognitiva processer i hjärnan, men deep learning visade att mönster kan läras direkt från data genom att prediktera nästa steg baserat på tidigare information (*The Cognitive Research behind AI's Rise*, 2024). Detta var en stor framgång för OpenAI, eftersom det möjliggjorde utvecklingen av avancerade generativa modeller som GPT (Generative Pre-trained Transformer) och DALL·E. GPT-modeller t.ex. är byggda på transformer-arkitekturen och använder deep learning-modeller såsom självuppmärksamhetsmekanismer (self-attention) och neurala nätverk för sekvensbearbetning.

Tabell 1 visar de största modellerna som finns idag inom artificiell intelligens och maskininlärning. Den visar att de flesta maskininlärningsuppgifter faller inom följande kategorier:

1. **Regression** – förutsäga kontinuerliga värden (t.ex. linjär regression, beslutsträd).
2. **Klassificering** – identifiera vilken kategori en datapunkt tillhör (t.ex. logistisk regression, SVM, Random Forest).
3. **Klustring** – gruppindelning av data utan fördefinierade etiketter (t.ex. k-means).
4. **Generativ modellering** – skapa nya data som liknar inlärningsdata (t.ex. GANs, VAEs).
5. **Sekvensmodellering** – analysera sekventiella data (t.ex. LSTM, Transformer).
6. **Bild- och språkbearbetning** – modellera och förstå bilder och språk (t.ex. CNN, BERT, ResNet).

Tabell 1. Centrala metoder inom maskin- och djupinlärning

Model	År	Utveklare	Typ	Uppgift	Referens
Linear Regression	1894	Sir Francis Galton	ML	Regression	(Galton, 1894)
Logistic Regression	1944	Joseph Berkson	ML	Classification	(Berkson, 1944)
Nearest Neighbour Algorithm (k-NN)	1951	Evelyn Fix; Joseph Hodges	ML	Classification & Reg.	(Hodges, 1951)
Perceptron	1957	Frank Rosenblatt	ML	Classification	(Rosenblatt, 1957)
k-Means	1957	Steinhaus; Lloyd; Forgy	ML	Clustering	(Forgy, 1965; Steinhaus, 1957)
Naive Bayes	1961	Reverend Thomas Bayes (1760); Maron (1961)	ML	Classification	(Maron, 1961)
Decision Trees	1963	Morgan; Sonquist	ML	Classification & Reg.	(Morgan & Sonquist, 1963)
Hopfield Network	1982	John Hopfield	NN / ML	Associative Memory	(Hopfield, 1982)
Classification and Regression Trees	1984	Breiman et al.	ML	Classification & Reg.	(Breiman et al., 1984)
Boltzmann Machine	1985	Ackley, Hinton, Sejnowski	ML / DL	Generative Modeling	(Ackley et al., 1985)
Backpropagation	1986	Rumelhart, Hinton, Williams	ML / DL	NNs (Class/Reg)	(Rumelhart et al., 1986)
Convolutional Neural Network	1989	Yann LeCun et al.	DL	Classification	(LeCun et al., 1989)
Support Vector Machines	1996	Vapnik; Chervonenkis (1964)	ML	Classification & Reg.	(Vapnik et al., 1996)
Random Decision Forests	1995	Tin Kam Ho	ML	Classification & Reg.	(T. K. Ho, 1995)
Adaptive Boosting	1995	Yoav Freund; Robert Schapire	ML	Classification	(Freund & Schapire, 1995)
Long Short-Term Memory	1997	Sepp Hochreiter; Jürgen Schmidhuber	DL	Sequence prediction	(Hochreiter & Schmidhuber, 1997)
Gradient Boosting Machines	1999	Jerome Friedman	ML	Classification & Reg.	(Friedman, 2001)
Deep Belief Networks	2006	Geoffrey Hinton	DL	Unsupervised Pretrain	(Hinton et al., 2006)
AlexNet	2012	Krizhevsky, Sutskever, Hinton	DL	Image Classification	(Krizhevsky et al., 2017)
Generative Adversarial Networks	2014	Ian Goodfellow et al.	DL	Generative modeling	(Goodfellow et al., 2014)
Attention Mechanism	2014	Dzmitry Bahdanau et al.	DL	Sequence modeling	(Bahdanau et al., 2014)
XGBoost	2016	Tianqi Chen & Carlos Guestrin	ML	Classification & Reg.	(Chen & Guestrin, 2016)
Light Gradient-Boosting Machine	2016	Guolin Ke et al.	ML	Classification & Reg.	(Ke et al., 2017)
ResNet	2016	He et al.	DL	Image Classification	(He et al., 2016)
AlphaGo	2016	DeepMind (David Silver et al.)	RL / DL	Game Playing (Go)	(Silver et al., 2016)
Transformer	2017	Ashish Vaswani et al.	DL	Sequence modeling	(Vaswani et al., 2017)
CatBoost	2017	Prokhorenkova, Gulin et al.	ML	Classification & Reg.	(Prokhorenkova et al., 2017)
BERT	2018	Jacob Devlin et al.	DL	Language Modeling	(Devlin et al., 2019)
GPT (Generative Pre-trained Transformer)	2018	OpenAI (Alec Radford et al.)	DL	Language Modeling	(Radford & Narasimhan, 2018)
Vision Transformer (ViT)	2020	Dosovitskiy et al.	DL	Image Classification	(Dosovitskiy et al., 2020)
Diffusion Models (DDPM)	2020	Jonathan Ho et al.	DL	Generative Modeling	(J. Ho et al., 2020)

1.2 OHR, OCR, Computer Vision

Datorer lagrar och bearbetar all data i form av nollor och ettor, därför att de är byggda på digital elektronik som använder binära transistorer. T.ex. fiberoptik som är dagens informationsbärare, använder ljuspulser för att representera data, där ljus "på" och "av" motsvarar binära ettor och nollor.

Computer Vision utvecklades från ett behov att automatisera visuell perception, såsom objektigenkänning, bildanalys och tolkning av visuella data i realtid. För att kunna bearbeta pixlar och visuella data omvandlar datorer bilder till numeriska matriser, där varje pixel representeras av binära värden eller intensitetsnivåer. Därför kan vi säga att computer vision är ett område inom AI som gör det möjligt för datorer att förstå visuella data från bilder eller videor. Computer Vision blev väldigt populärt med framstegen inom deep learning, särskilt med utvecklingen av Convolutional Neural Networks (CNN) av Yann LeCun och hans kollegor (LeCun et al., 1990).

I Computer Vision står OHR för Optical Handwriting Recognition, vilket är tekniken för att automatiskt identifiera och tolka handskriven text från bilder eller dokument. Apple Newton (1993) var en tidig PDA (Personal Digital Assistant) som använde handskriftigenkänning och var en föregångare till dagens smartphones och surfplattor (Wikipedia contributors, 2025). Med förbättringar inom artificiella neurala nätverk (ANN) började forskare använda algoritmer för att identifiera mönster i handskrift. Google Handwriting Input och Microsoft Ink Recognition är exempel på moderna OHR-system. Det används i elektroniska signaturer, juridiska dokument och digitala anteckningar. OHR används i säkerhetssystem för att känna igen en individs unika skrivstil.

Nära OHR (Optical Handwriting Recognition) står OCR (Optical Character Recognition), som används för att känna igen tryckt text istället för handskriven text. Den första kommersiella OCR-maskinen för att digitalisera tryckta böcker och tidningar, kallad "Reading Machine", utvecklades 1974 av Ray Kurzweil (Craine, 2009).

MNIST-datasetet är känt som en standard inom maskininlärning och innehåller handskrivna siffror (0–9) som används för att träna och testa bildigenkänningsmodeller, särskilt inom deep learning. MNIST-datasetet skapades av Yann LeCun och Corinna Cortes (LeCun & Cortes, 2005) genom att kombinera och bearbeta handskrivna siffror från NIST-datasetet. Bokstaven M i MNIST står för Modified, eftersom datasetet är en modifierad version av det ursprungliga NIST-datasetet. Datan är modified på så sätt att bilderna i MNIST-datasetet har normaliserats till enhetlig storlek (28x28 pixlar), centrerats och förbehandlats för att vara mer lämpliga för träning av maskininlärningsmodeller.

Uppgiften för modellen är alltså att klassificera handskrivna siffror (0–9). Bland de modeller som används för klassificering finns logistisk regression, stödvektormaskiner (SVM), k-NN, beslutsträd, Random Forest, neurala nätverk och Convolutional Neural Networks (CNN). Eftersom vi har begränsad tid för det här projektet och mycket tillgänglig information om vilka modeller som presterar bäst, ska vi testa bara tre: **Logistisk Regression, Support Vector Machines (SVM) och Convolutional Neural Networks (CNN)**.

1.3 Logistic regression

Logistisk regression utvecklades först för att **modellera sannolikheten för binära utfall** inom statistik och biomedicinska studier. Den modellen använder **sigmoida funktionen** för att omvandla insignalvärden till sannolikheter mellan 0 och 1, vilket gör den lämplig för binär klassificering.

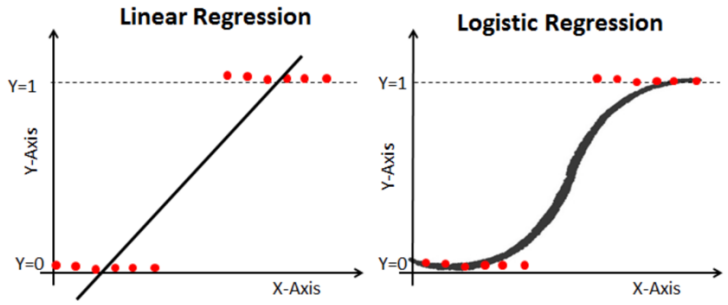

Bild 1. Linjär vs Logistisk regression (Navlani, 2019)

Som vi kan se från bilden 1 - jämförs linjär regression och logistisk regression:

Linjär regression - här dras en rak linje genom datapunkterna. Modellen försöker förutsäga ett värde längs den linjen, vilket fungerar bra för kontinuerliga resultat (t.ex. att förutsäga temperatur eller huspriser). Men för klassificering, där vi vill ha antingen 0 eller 1, blir resultatet problematiskt eftersom linjen kan ge värden större än 1 eller mindre än 0.

Logistisk regression - istället för en rak linje använder logistisk regression en sigmoidkurva (S-formad). Den ser till att alla förutsägelser ligger mellan 0 och 1, vilket gör den användbar för binär klassificering (t.ex. "Ja eller Nej", "Sant eller Falskt"). Ju närmare en punkt är 1, desto högre sannolikhet att den tillhör kategori 1, och ju närmare 0, desto mer sannolikhet att den tillhör kategori 0.

Det kan vara missvisande att den logistiska regressionsmodellen kallas för "regression", eftersom den egentligen används för klassificering snarare än för att förutsäga kontinuerliga värden, vilket är vad regression vanligtvis gör. Anledningen till namnet är att modellen bygger på en regressionsmetod, men istället för att förutsäga numeriska värden omvandlar den resultatet till en sannolikhet mellan 0 och 1 med hjälp av sigmoidfunktionen. Därefter används en tröskel (t.ex. 0,5) för att avgöra om utfallet ska klassas som kategori 0 eller 1.

Sigmoid funktion svarar på frågan "Är detta sant eller falskt?". Medan Softmax svarar på frågan "Vilken av dessa flera möjligheter är mest sannolik?".

1.4 Support Vector Machines

Support Vector Machine (SVM) tar sitt namn från **supportvektorer**, som är de datapunkter närmast beslutsgränsen och används för att definiera den optimala separationslinjen.

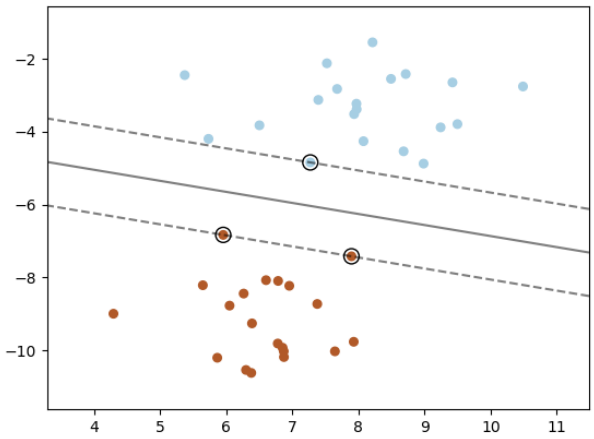

Bild 2. SVM princip (Nguyen et al., 2023)

Bilden 2 visar en SVM-klassificering med en beslutsgräns (solid linje) och marginaler (streckade linjer). De cirklade punkterna är supportvektorer, vilka avgör den optimala separationslinjen mellan de två klasserna. Den modellen är mest användbar för högdimensionell data, textklassificering, bildigenkänning och biometrisk identifiering, där den kan hantera komplexa mönster. Idag är SVM fortfarande användbart, men deep learning har ersatt det i många uppgifter som t.ex. bild- och språkigenkänning.

1.5 Convolutional Neural Networks

För att förstå neurala nätverk kan man tänka sig ett team av experter som förfinar information steg för steg. En mänsklig neuron fungerar som en signalprocessor, tar emot signaler, tillämpar en tröskel och skickar en signal. Tröskel i det här fallet betyder en gräns för aktivering, där en neuron endast skickar en signal om insignalens värde överstiger denna gräns. På samma sätt tar en artificiell neuron emot signaler, applicerar vikter, passerar dem genom en aktiveringsfunktion och skickar en signal. Några vanliga aktiveringsfunktioner inkluderar sigmoid, tanh och ReLU. Neuroner kan vara kopplade till varandra i en synaps. En synaps i artificiella neurala nätverk motsvarar en viktad koppling mellan neuroner som avgör signalens styrka och påverkar inlärning. Neuroner bildar nätverk för ökad kapacitet (perceptron). I en perceptron får alla neuroner samma input. En enkel perceptron har svag inlärningsförmåga, men fler lager mellan in- och utgång, kallade dolda lager, gör den starkare. Flerskiktsperceptron tränas med backpropagation.

Vikter bestäms genom träning med backpropagation och optimeringsalgoritmer som gradientnedstigning. Mycket förenklat är backpropagation en inlärningsprocess där nätverket justerar vikter genom att sprida fel bakåt och minimera skillnaden mellan förutsägelse och verklighet. CNN använder konvolution istället för allmän matris-multiplikation i minst ett av sina lager. Konvolution kan visualiseras som ett filter som rör sig över en bild och beräknar nya värden baserat på närliggande pixlar. Det finns två vanliga pooling-typer: maxpooling (tar det högsta värdet i feature-mappen) och medelpooling (tar medelvärdet). Konvolutions- och poolinglager används för att extrahera egenskaper. Efter detta krävs ett fullständigt anslutet lager för klassificering, likt en flerskiktsperceptron, där den extraherade informationen kopplas till utgångslagret och input klassificeras.

En viktig person som bidrog till backpropagation är Geoffrey Hinton, som fick Turingpriset 2018 för sitt arbete inom djupinlärning (*Fathers of the Deep Learning Revolution Receive ACM A.M. Turing Award*, 2018). Ordet convolutional betyder "vikning" eller "sammanslagning" och används i convolutional neural networks (CNNs) för att beskriva hur filter appliceras på data, särskilt bilder, för att extrahera mönster genom att kombinera närliggande pixlar. Vanliga neurala nätverk presterar sämre på bild- och mönsterigenkänning eftersom de inte bevarar spatiala relationer och kräver fler parametrar. En modell som presterar bättre än CNN på vissa uppgifter är transformers, särskilt Vision Transformers (ViT), som använder självuppmärksamhet för att hantera globala relationer i data (Dosovitskiy et al., 2020).

2 Metod

2.1 Setup och modellutvärdering

Även om jag lärt mig från tidigare nämnda källor att CNN presterar bäst på bildigenkänning, tränar jag ändå andra modeller såsom logistisk regression och SVM för jämförelse utvärdering av prestandaskillnader.

Först och främst skapar jag en virtual environment med Python 3.9 för att säkerställa kompatibilitet och undvika konflikter med systemets paket. Sedan skapar jag ett lokalt repo och kopplar det till en remote med Git.

Scikit-learn är det viktigaste biblioteket inom maskininlärning som erbjuder enkla verktyg för klassificering, regression, klustring och dataförbehandling. Andra paket som jag kommer att använda inkluderar NumPy för numeriska beräkningar, Matplotlib för visualisering och TensorFlow/Keras för djupa nätverk. Streamlit används för att bygga och distribuera interaktiva webbappar för dataanalys och maskininlärning.

Eftersom det är ett mindre projekt och jag är begränsad i tid, mäter jag prestanda endast med accuracy. Accuracy är dock inte alltid tillräcklig, särskilt vid obalanserade dataset, där andra mått som precision, recall och F1-score kan ge en mer rättvis bild av modellens prestanda.

Efter att jag laddat ner MNIST-data och sparade det som X, y, där X innehåller bilddata som numeriska matriser och y innehåller motsvarande etiketter för siffrorna (0–9), randomiserade jag datasetet och konverterade det till 2D från bildformat, för att anpassa formatet till modeller som kräver platta vektorer istället för bildmatriser. Jag applicerade även StandardScaler på X för att normalisera data, eftersom logistisk regression och SVM antar att funktioner är skalade för bättre prestanda och konvergens. Sedan splittrade jag datasetet i training, validation och test för att träna modellen, justera hyperparametrar med grid search och utvärdera prestandan på osedd data.

2.2 Optimering och träning av maskininlärningsmodeller

För Logistic Regression, optimerade jag regulariseringsstyrka - C (0.01, 0.1, 1), penalty (L1, L2) och konvergenstolerans - tol (0.01, 0.1, 1) med Grid Search för att förbättra modellens prestanda.

Eftersom SVM är beräkningsintensiv vid stora dataset, använde jag PCA som behåller 95% av varians för att påskynda träningen. PCA (Principal Component Analysis) är en dimensionreduktionsmetod som omvandlar data till ortogonala komponenter, bevarar maximal varians och minskar beräkningskomplexiteten. Sedan optimerade jag hyperparametrarna med Grid Search, där jag testade C (regulariseringsparameter: 1, 5, 10) och gamma (kernelkoefficient för RBF: 0.0001, 0.001, 0.01). RBF (Radial Basis Function) är en icke-linjär kärnfunktion som används i SVM för att kartlägga data till en högre dimension, vilket gör det möjligt att separera komplexa mönster (*Recognizing Hand-Written Digits*, n.d.).

CNN är inte lika beroende av normalisering som traditionella ML modeller, eftersom konvolutionella lager extraherar egenskaper direkt från rådata. Men man ska ändå normalisera MNIST-data genom att skala pixelvärden från 0–255 till 0–1, t.ex. med: X = X / 255.0. Annars är risken att modellen tränas långsammare, får sämre konvergens eller fastnar i lokala minima på grund av ojämna skalförhållanden i datan. 255 kommer från att pixelvärden i gråskalebilder representeras som heltal mellan 0 och 255, där 0 är svart och 255 är vitt.

När man tränar en modell i TensorFlow/Keras, specificerar man en sekventiell CNN-arkitektur med konvolutionslager, pooling, och dense-lager, kompilerar den med Adam-optimerare och cross-entropy-loss, och tränar den med fit() på tränings- och valideringsdata. En sekventiell CNN-arkitektur är en av de vanligaste och mest effektiva modellerna för bildigenkänning, där lager läggs på varandra i ordning för att extrahera och klassificera bildegenskaper. Andra vanliga arkitekturer för bildigenkänning är ResNet (med restkopplingar), VGG (djupa, enkla lager), Inception (med parallella filter) och Vision Transformers (ViT) som bygger på självuppmärksamhet istället för konvolutioner. Pooling är en operation som minskar dimensionen på feature maps genom att sammanfatta information, ofta med max- eller medelvärde. Ett dense-lager är ett fullt anslutet lager där varje neuron tar emot input från alla neuroner i föregående lager. Adam-optimerare är en gradientnedstigningsalgoritm som kombinerar momentum och adaptiv inlärningshastighet. Cross-entropy-loss är en förlustfunktion som mäter skillnaden mellan de förutsagda sannolikhetsfördelningarna och de faktiska etiketterna, vanligtvis använd för klassificeringsproblem.

2.3 Generering av syntetisk data och implementering av applikationer

Jag skapade syntetisk data genom att generera handritade "+" och "-" symboler med varierande tjocklek, längd, jitter och rotation, sparade dem som 28x28 bilder, och lade till dem i MNIST-datasetet för att träna modellen i aritmetiska operationer. Jag augmenterade den nya datasetet samt MNIST genom att normalisera bilder, applicera dataaugmentation (såsom rotation och jitter) och sedan batcha datan, för att förbättra modellens generaliseringsförmåga.

Jag sparade den tränade CNN-modellen och byggde en Tkinter GUI-app där användaren kan rita siffror och tecken, modellen segmenterar och klassificerar dem, och evaluerar uttrycket för att lösa enkla aritmetiska problem. Jag migrerade från Tkinter GUI till Streamlit genom att använda st_canvas för ritning, implementera bildsegmentering och modellprediktion, och skapa en webbaserad handritad ekvationslösare som tolkar och beräknar uttryck i realtid.

3 Resultat och Diskussion

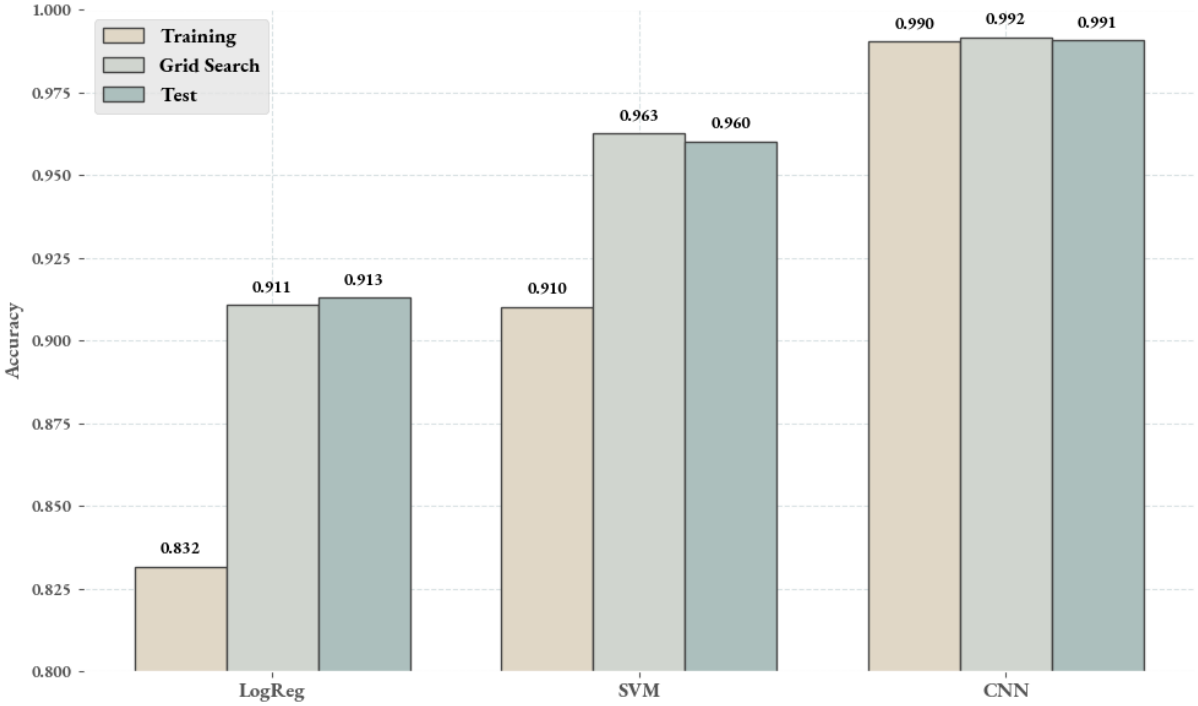

Bild 3. Jämförelse av modellernas accuracy på MNIST data.

Bilden 3 visar jämförelse av training, Grid Search och testaccuracy för Logistisk Regression (LogReg), Support Vector Machine (SVM) och Convolutional Neural Network (CNN) på MNIST-datasetet. CNN har högst accuracy (~99%) på alla steg, medan SVM förbättras markant genom Grid Search (från 0.910 till 0.963). LogReg har lägst prestanda men förbättras något med Grid Search. Trenden visar att mer komplexa modeller och hyperparametertuning leder till bättre resultat.

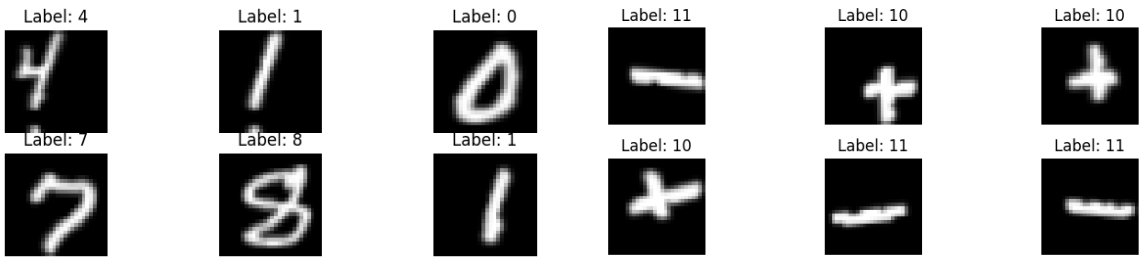

Bild 4. Augmenterade + och – symboler samt siffror från MNIST-datasetet

Bilden 4 visar ett resultat av augmentering av både syntetiska “+” och “-” samt MNIST-data, där variation i handstil, rotation och brus har introducerats för att öka modellens generaliseringsförmåga. En viktig detalj att notera är att "+" och "-" tecken har label 10 och 11, eftersom CNN inte kan hantera symboler som klassnamn och därför kräver numeriska etiketter för klassificering.

När en användare skriver en räkneformel, segmenterar algoritmen varje siffra och symbol genom att binarisera bilden, hitta konturer, extrahera och sortera tecken, samt centrera dem i 28×28 bilder för att matcha CNN-modellens indata.

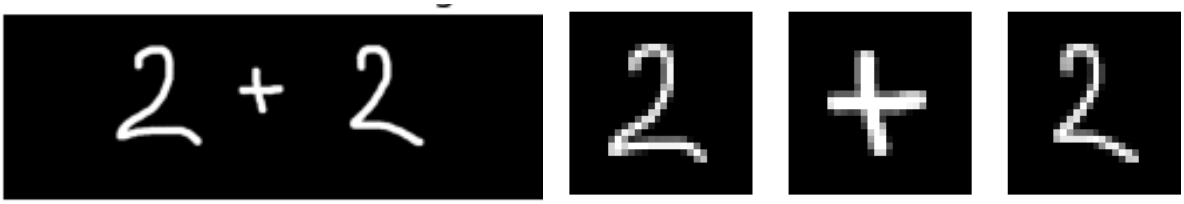

Bild 5. Segmentering

Bilden 5 visar hur räkneformeln ser ut före och efter segmentering, där hela uttrycket först fångas som en sammanhängande bild och sedan delas upp i enskilda siffror och symboler, centrerade i 28×28 bilder för vidare klassificering av CNN-modellen.

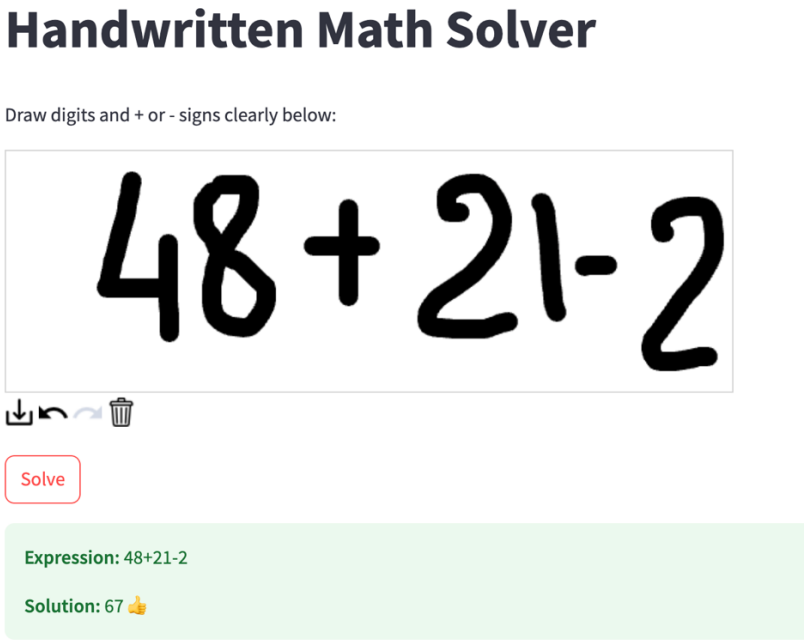

Bild 6. Handritad matematiklösare i Streamlit – exempel på beräkning

Bilden 6 visar den slutliga applikationen i Streamlit, där en användare har skrivit ett handritat matematiskt uttryck "48 + 21 - 2". Applikationen har korrekt segmenterat, tolkat och beräknat uttrycket, vilket resulterar i lösningen 67, som visas i grön ruta längst ner. Den är tillgänglig på eq-solver.streamlit.app, där användare kan rita matematiska uttryck och få realtidsprognoser och lösningar.

4 Slutsats

Detta projekt syftade till att utveckla en webbaserad applikation för handskriven matematisk igenkänning, där användare kan skriva räkneuppgifter för hand och få dem tolkade och beräknade i realtid. Genom en systematisk metod bestående av litteratursökning, modellutvärdering, datasetförbättring, hyperparametertuning och applikationsutveckling har **målet uppnåtts**.

5 Teoretiska frågor

För att ta projektet vidare kan man utöka funktionaliteten genom att lägga till fler matematiska operatorer som multiplikation och division. Segmenteringsalgoritmen kan förbättras för att hantera överlappande eller sammanhängande siffror mer exakt. Vidare kan transformerbaserade modeller testas för att utvärdera om de presterar bättre än CNN. Ett API skulle möjliggöra integration med andra applikationer. Slutligen kan systemet anpassas för att känna igen matematiska uttryck på olika språk och notationer.

6 Självutvärdering

Projektet uppfyller kraven för "godkänd" och går längre genom att använda flera maskininlärningsmetoder. Logistisk regression, SVM och CNN har jämförts, optimerats och utvärderats. Modellval och resultat har redovisats och diskuterats kritiskt, vilket uppfyller kraven för betyget väl godkänd.

7 Källförteckning

Ackley, D., Hinton, G., & Sejnowski, T. (1985). A learning algorithm for boltzmann machines. *Cognitive Science*, *9*(1), 147–169. https://doi.org/10.1016/s0364-0213(85)80012-4

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. In *arXiv [cs.CL]*. arXiv. http://arxiv.org/abs/1409.0473

Berkson, J. (1944). Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, *39*(227), 357–365. https://doi.org/10.1080/01621459.1944.10500699

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification And Regression Trees* (1st Edition). Routledge. https://doi.org/10.1201/9781315139470

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *arXiv [cs.LG]*. arXiv. http://arxiv.org/abs/1603.02754

Craine, A. G. (2009, July 15). *Ray Kurzweil*. Encyclopedia Britannica. https://www.britannica.com/biography/Raymond-Kurzweil

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/n19-1423

Djupinlärning jämfört med maskininlärning - Azure Machine Learning. (2024, September 2). https://learn.microsoft.com/sv-se/azure/machine-learning/concept-deep-learning-vs-machine-learning?view=azureml-api-2

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, *abs/2010.11929*. https://doi.org/10.11929/1000

Fathers of the Deep Learning Revolution Receive ACM A.M. Turing Award. (2018). https://awards.acm.org/about/2018-turing

Forgy, E. W. (1965). Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, *21*(3), 761–777. https://www.jstor.org/stable/2528559

Freund, Y., & Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In *Lecture Notes in Computer Science* (pp. 23–37). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-59119-2_166

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, *29*(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Galton, F. (1894). *Natural Inheritance*. Macmillan and Company.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Networks. In *arXiv [stat.ML]*. arXiv. http://arxiv.org/abs/1406.2661

He, K., Zhang, X., Ren, S., & Sun, J. (2016, June). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA. https://doi.org/10.1109/cvpr.2016.90

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. *Neural Information Processing Systems*, *abs/2006.11239*, 6840–6851. https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html

Ho, T. K. (1995). Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*. 3rd International Conference on Document Analysis and Recognition, Montreal, Que., Canada. https://doi.org/10.1109/ICDAR.1995.598994

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hodges, E. F. A. (1951). *Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties*. https://www.scirp.org/reference/referencespapers?referenceid=1603038

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, *79*(8), 2554–2558. https://doi.org/10.1073/pnas.79.8.2554

Introduction to Machine Learning. (2003, May 25). *Machine learning*. Wikimedia Foundation, Inc. https://en.wikipedia.org/wiki/Machine_learning

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient Gradient Boosting Decision Tree. *Neural Information Processing Systems*, 3146–3154. https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541–551. https://doi.org/10.1162/neco.1989.1.4.541

LeCun, Y., & Cortes, C. (2005). *The mnist database of handwritten digits*. https://www.semanticscholar.org/paper/The-mnist-database-of-handwritten-digits-LeCun-Cortes/dc52d1ede1b90bf9d296bc5b34c9310b7eaa99a2

LeCun, Y., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jacke, L. (1990). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *2*. https://scholar.google.com/citations?user=WLN3QrAAAAAJ&hl=en&oi=sra

Maron, M. E. (1961). Automatic indexing: An experimental inquiry. *Journal of the ACM*, *8*(3), 404–417. https://doi.org/10.1145/321075.321084

Morgan, J. N., & Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, *58*(302), 415–434. https://doi.org/10.1080/01621459.1963.10500855

Navlani, A. (2019, December 16). *Python Logistic Regression Tutorial with Sklearn & Scikit*. DataCamp. https://www.datacamp.com/tutorial/understanding-logistic-regression-python

Nguyen, H., Di Troia, F., Ishigaki, G., & Stamp, M. (2023, February 1). *SVM separating hyperplane [46]*. ResearchGate. https://www.researchgate.net/figure/SVM-separating-hyperplane46_fig1_368474711

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2017). CatBoost: unbiased boosting with categorical features. In *arXiv [cs.LG]*. arXiv. http://arxiv.org/abs/1706.09516

Radford, A., & Narasimhan, K. (2018). *Improving language understanding by generative pre-training*. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

Recognizing hand-written digits. (n.d.). Scikit-Learn. Retrieved March 21, 2025, from https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory. https://books.google.com/books/about/The_Perceptron_a_Perceiving_and_Recogniz.html?hl=sv&id=P_XGPgAACAAJ

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489. https://doi.org/10.1038/nature16961

Steinhaus, H. (1957). *Sur la division des corps matériels en parties*. https://zbmath.org/?format=complete&q=an:0079.16403

The cognitive research behind AI's rise. (2024, November 18). https://news.stanford.edu/stories/2024/11/from-brain-to-machine-the-unexpected-journey-of-neural-networks

Vapnik, V., Golowich, S., & Smola, A. (1996). Support Vector method for function approximation, regression estimation and signal processing. *Neural Information Processing Systems*, 281–287. https://proceedings.neurips.cc/paper_files/paper/1996/file/4f284803bd0966cc24fa8683a34afc6e-Paper.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *arXiv [cs.CL]*. arXiv. http://arxiv.org/abs/1706.03762

Wheeler, J. (2025, January 6). *ANI vs. AGI vs. ASI: What They Mean and Why 2025 Could Be the Year of AGI*. Virtually Caffeinated. https://www.virtuallycaffeinated.com/2025/01/06/ani-vs-agi-vs-asi-what-they-mean-and-why-2025-could-be-the-year-of-agi/

Wikipedia contributors. (2025, March 18). *Apple Newton*. Wikipedia, The Free Encyclopedia; Wikimedia Foundation, Inc. https://en.wikipedia.org/wiki/Apple_Newton