

Studieretningsprojekt 2022/2023

3y 05 Christian Egelund Hansen	2020/21 MAA-FYA-KeB
Fag i StudieRetningsProjektet	Vejledere
Matematik A	Søren Bredlund Caspersen SC@sctknud-gym.dk
Informatik C	Heidi Skovbak Graversen hg@sctknud-gym.dk
Område for StudieRetningsProjektet:	Hvordan en computer lærer vha kunstig intelligens

Hvordan kan man udnytte kunstig intelligens i forskellige industrier?

Redegør kort for matricer og matrixmultiplikation og transponering af en matrix. Formuler og bevis sætningen

$$(AB)^t = B^t A^t,$$

hvor t betegner transponering.

Generaliser til 3. ordens Tensor-matematik, og redegør for, hvordan en convolution-operation virker med en 2×2 kernel. Løs opgaven vedlagt på bilag 1.

Redegør for, hvordan man kan implementere samme convolution-operation i et program skrevet i fx *Python*.

Redegør kort for, hvordan *Convolutional Neural Network*, CNN adskiller sig fra et klassisk *Neural Network*.

Du skal anvende og analysere selvvalgt kode, som du kan anvende til at træne en kunstig intelligens i at genkende cifrene fra 0 til 9 vha. et bibliotek som implementerer CNN. Anvend programmet på billedfilen i bilag 2.

Diskuter, hvor CNN med fordel kan anvendes, og diskuter fordele og ulemper ved implementering og brug af kunstig intelligens.

Bilag 1: Opgave med Convolution
Bilag 2: Billede til ciffergenkendelse

Opgavens omfang: 15-20 sider.



Resume

I mit projekt undersøges der, hvordan kunstig intelligens, mere specifikt machine learning, virker. For at forstå matematikken bag machine learning kræver det viden om matricer, tensor og convolution-operator, hvilket redegøres for. Årsagen til, at denne matematik er nødvendig, er at inputtet enten er matricer eller tensor. Convolution-operator er også vigtigt, da det viser, hvordan et convolutional neural network fungerer. Med den matematiske viden om overstående, kan man implementere logikken bag convolutional-operator i et python program. Programmet kan udregne en convolution mellem et input og en kernel. Opgaven på bilag 1 udregnes både matematisk og vha. programmet, hvor resultatet = $\begin{bmatrix} 20 & 47 & 40 \\ 38 & 48 & 36 \end{bmatrix}$. Efterfølgende redegøres for neural network og convolutional neural network. Her forklares det, at convolutional neural network bliver anvendt i billede-genkendelse fremfor neural network, da et convolutional neural network giver et bedre og hurtigere svar. Dette sker, da convolutional neural network skal forholde sig til mindre vægte w end et neural network. Derudover analyseres koden bag et convolutional neural network, der er programmeret. Her forklares det, hvordan bestemte linjer kode virker og hvorfor de anvendes. Dette program kan genkende cifre fra 0 til 9. Programmet blev anvendt på bilag 2 og kunne identificere cifrene 1 og 3. Programmet er altså vellykket. Der bliver diskuteret, hvor man med fordel kan anvende kunstig intelligens. Her argumenteres for, at man med fordel kan anvende kunstig intelligens i sundhedssektoren, da det vil spare tid. Der diskuteses også fordele og ulemper ved kunstig intelligens. Fordelene er f.eks., at man kan hjælpe mulige kræftpatienter ved at identificere tumorer. En af ulemperne ved kunstig intelligens er den markante mængde af data, som kunstig intelligens kræver, kan hackes og udnyttes. Til sidst konkluderes.

Indhold

Indledning	4
Matricer, matrixmultiplikation og transponering	4
Matricer	4
Matrixmultiplikation	5
Transponering	6
Tensor-matematik og convolution operation	7
Tensor-matematik	7
Convolution-operation	9
Implementering af convolution-operation i <i>Python</i>	10
Neural network og convolutional neural network, <i>CNN</i>	13
Neural network	13
Convolutional neural network, <i>CNN</i>	14
”Hello world”i machine learning	16
Fordeler og ulemper af kunstig intelligens	21
Konklusion	22
Litteraturliste	24
Bilag	26
Bilag 1	26
Bilag 2	27

Indledning

Er du bange for, at robotterne tager over? Dette er et spørgsmål mange har stillet sig selv efter at have set store Hollywood-film, som Terminator eller The Matrix. Men hvordan virker denne kunstige intelligens, som robotterne bliver styret af og er den virkelig så farlig, som vi tror? For at man kan forholde sig til dette, er det vigtigt at forstå, hvad kunstig intelligens overhovedet er, hvordan det virker og hvordan den bruges?

For at kunne svare på projektets problemformulering gives først en redegørelse for matricer og nogle af de regneregler, som er bestemte for matricer. I forlængelse af dette redegøres der yderligere for vigtigheden af, hvorfor matricer og tensorer er essentielle i forhold til convolution-operation. Dette vil lede videre til et program skrevet i *Python*, som kan udregne convolution-operation. Herefter beskrives, hvad et neural network er. Med en viden om convolution-operation og neural network kan det forklares, hvad et convolutional neural network er og hvordan det er anderledes fra et neural network. Dette vil lede frem til, hvordan man programmerer et convolutional neural network og om det fungerer. Det diskutes, i hvilke sammenhænge kunstig intelligens med fordel kan benyttes og i hvilke sammenhænge, det er mere problematisk og kan lede til debat. Til sidst konkluderes ud fra ovenstående.

Matricer, matrixmultiplikation og transponering

I det følgende afsnit, vil jeg redegøre for matricer, matrixmultiplikation og transponering af matricer. Hvorefter jeg vil formulere og bevise sætningen

$$(AB)^t = B^t A^t$$

hvor t betegner transponering.

Matricer

En matrix er et talskema, hvor tallene er organiseret i rækker i og søjler j . En matrix \mathbf{M} med rækker i og søjler j betegnes $(i \times j)$ -matrix, hvilket beskriver matrixens dimensioner og er vist

herunder.

$$M = \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} & \dots & m_{1,j-1} & m_{1,j} \\ m_{2,1} & m_{2,2} & m_{2,3} & \dots & m_{2,j-1} & m_{2,j} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ m_{i-1,1} & m_{i-1,2} & m_{i-1,3} & \dots & m_{i-1,j-1} & m_{i-1,j} \\ m_{i,1} & m_{i,2} & m_{i,3} & \dots & m_{i,j-1} & m_{i,j} \end{pmatrix}$$

Tallene i matrixen kaldes matrixens elementer, hvor det enkelte tal betegnes m_{ij} og bliver kaldt det ij 'te element. Man betegner altså matricer $M \in \mathbb{R}^{m \times n}$, hvor m og n er antallet af rækker og søjler. Man betegner vektorer med en søjle og flere rækker, derfor kan man også betegne en vektor med størrelsen k , som en $(k \times 1)$ -matrix. Dette betyder, at en vektor er en 1-dimensionel matrix $\mathbf{V} \in \mathbb{R}^{m^1}$.

Matrixmultiplikation

Når man ganger to matricer sammen, kaldes det matrixmultiplikation. Hvis man har to matricer $A_{m \times n}$ og $B_{n \times p}$, hvor A har samme antal rækker n , som B har antal søjler n , kan man multiplicere dem, hvorefter man får en ny $(m \times p)$ -matrix C . Man kan beregne elementerne c_{ij} efter formlen².

$$c_{ij} = \sum_{k=1}^n a_{i,k} b_{k,j} = a_{i,1} b_{1,j} + a_{i,2} b_{2,j} + \dots + a_{i,n} b_{n,j}$$

Et eksempel på matrixmultiplikation er med en (2×2) -matrix A og (2×3) -matrix B .

$$C = A \cdot B = \begin{pmatrix} 3 & 2 \\ 5 & 2 \end{pmatrix} \cdot \begin{pmatrix} -1 & 6 & 3 \\ 4 & -3 & 3 \end{pmatrix} \quad (1)$$

$$= \begin{pmatrix} 3 \cdot (-1) + 2 \cdot 4 & 3 \cdot 6 + 2 \cdot (-3) & 3 \cdot 3 + 2 \cdot 3 \\ 5 \cdot (-1) + 2 \cdot 4 & 5 \cdot 6 + 2 \cdot (-3) & 5 \cdot 3 + 2 \cdot 3 \end{pmatrix} \quad (2)$$

$$= \begin{pmatrix} 5 & 12 & 15 \\ 3 & 24 & 21 \end{pmatrix} \quad (3)$$

Når man arbejder med matricer gælder det, at addition er associative og kommutative. Ved matrixmultiplikation gælder dette ikke. Matrixmultiplikation er derfor ikke-associative og ikke-kommutativ, dvs. at rækkefølgen af ens regneudtryk er vigtigt. For eksempel er $5 \times 3 = 3 \times 5$,

¹Torfi, Amirsina, 2020, Practical linear algebra for machine learning

²Ridder Ebbesen, Grete, Matricer og lineære ligningssystemer

hvor $A \cdot B \neq B \cdot A$ ³. I ovenstående eksempel gav AB en (2×3) -matrix, hvor BA ikke ville kunne udregnes, da antallet af rækker og søjler ikke er ens $n \neq n$.

Transponering

Matricer har også den funktion, at de kan transponeres. Ved transponering af en matrix bliver der dannet en ny matrix, som bytter om på den tidligere matrix's rækker i og søjler j . Når en matrix transponeres, betegner man matrixen med et t , den transponerede $(m \times n)$ -matrix $A = (a_{ij})$ bliver altså til $(n \times m)$ -matrix $A^t = (a_{ji})$, hvilket ses herunder⁴.

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}, \quad A^t = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{pmatrix}$$

Der opstår forskellige regneregler ved transponering, hvor $(AB)^t = B^t \cdot A^t$ er en af de mest relevante. Denne regneregel formulerer, at når to matricer, $(m \times n)$ -matrix A og $(n \times p)$ -matrix B , er ganget sammen og derefter transponeres, vil matrix B^t ganget med matrix A^t være lig med hinanden. Dette kan nu bevises med den tidlige teori, hvis man antager, at man har to matricer $(m \times n)$ -matrix A og $(n \times p)$ -matrix B .

Sætning:

$$(AB)^t = B^t \cdot A^t$$

Husker at

$$A = (a_{i,j}) \quad B = (b_{i,j}) \quad A^t = (a_{j,i}) \quad B^t = (b_{j,i})$$

beviser sætningen:

$$((AB)^t)_{i,j} = (AB)_{j,i} = \sum_{k=1}^n A_{j,k} \cdot B_{k,i} \tag{1}$$

$$= \sum_{k=1}^n (A^t)_{k,j} \cdot (B^t)_{i,k} \tag{2}$$

$$= \sum_{k=1}^n (B^t)_{i,k} \cdot (A^t)_{k,j} \tag{3}$$

$$= (B^t \cdot A^t)_{i,j} \tag{4}$$

³Wikipedia, Matrix, besøgt d. 21/3/2023

⁴Dawani, Jay, 2020, *Hands-on mathematics for deep learning*

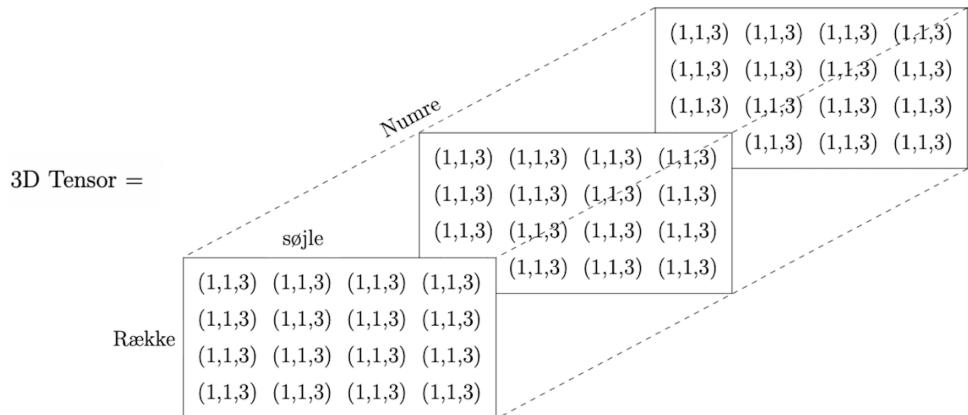
Man kan se, at matricerne $(AB)^t$ og $B^t A^t$ har samme dimensioner $m \times p$, derudover har elementerne i begge matricer betegnelsen $i j$. Da begge matricer har samme rækker, søjler og elementer, hvilket betyder de er ens, er sætningen $(AB)^t = B^t A^t$ nu bevist⁵.

Tensor-matematik og convolution operation

Tensor-matematik

En *tensor* er en N-dimensionel matrix. Dette er vigtig, da tensors er en af de største byggeblokke bag machine learning. Når man skal repræsentere numerisk data i machine learning, præsenterer man denne data med tensors. Dette betyder en tensor kan indeholde en eller flere matricer og derfor mere data. Dette kan ses herunder:

Figur 1: Inspiration fra stackexchange.com, link:<https://tex.stackexchange.com/questions/300109/simple-visualization-of-3d-matrix>



Tidligere er matricer beskrevet, som enten en 1-dimensionel matrix/vector eller en 2-dimensionel matrix. Da en tensor er en matrix med N-dimensionel data, kan man derfor beskrive denne 1-dimensionelle matrix/vector som en 1-dimensionel tensor, ligeledes med en 2-dimensionel matrix⁶.

$$\text{1D Tensor/vector} = \begin{bmatrix} 1 \\ 6 \\ 9 \end{bmatrix}, \quad \text{2D Tensor} = \begin{bmatrix} 8 & 29 & 13 & 2,9 \\ 9 & 4 & 10 & -1 \\ -0,2 & 3 & 4,4 & -10 \\ -1,2 & 8,7 & 31 & 1 \end{bmatrix}$$

⁵University of Manchester, Prerequisite Material

⁶stackexchange.com

link:<https://tex.stackexchange.com/questions/300109/simple-visualization-of-3d-matrix>

Man kan overgå 1 eller 2-dimensionelle matricer og tilføje N -dimensioner til en tensor. Dette er især vigtigt i machine learning, da data er opbevaret og repræsenteret gennem 3-dimensionelle/3. ordens tensor. Det vil sige, at man kan generalisere konceptet af en tensors fra matricer. Hvor man betegnede en vektor $v \in \mathbb{R}^m$ og en matrix $M \in \mathbb{R}^{m \times n}$, betegner man f.eks. en 3. ordens tensor $t \in \mathbb{R}^{m \times n \times d}$. Her betegner man de enkelte elementer i tensoren, som det ijd 'te element⁷.

Årsagen til at 3. ordens tensorer er essentielle

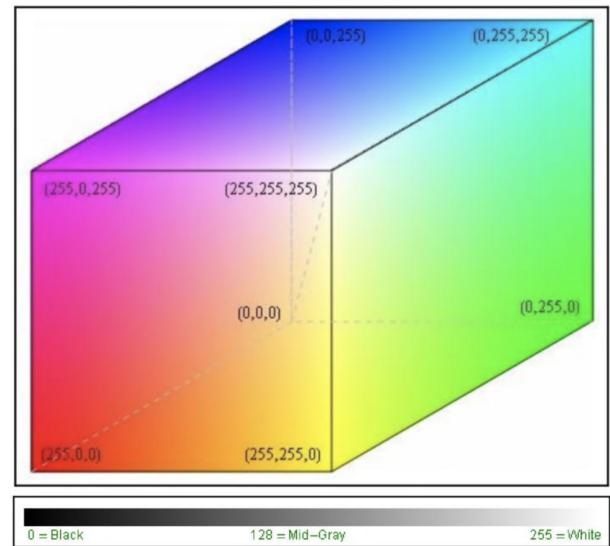
i machine learning og specifikt i *convolutional neural networks, CNN*, er den måde convolutional neural networks genkender f.eks billeder. Et billede kan enten være grayscale(sort og hvidt) eller farvet(RGB), som er lavet af pixels. Når et billede er grayscale, kan hvert individuelle pixel have en værdi mellem 0 – 255, hvilket kan ses nederst på figur 2.

Når et billede derimod har farver(RGB), har hvert pixel tre forskellige værdier fra farverne rød, grøn og blå, da alle farver kan kombineres fra disse grundfarver. Hver pixel kan have en værdi fra hver af de tre farver mellem 0 – 255. Således får man brug for en 3. ordens tensor til at opbevare data for hver pixel. Man kan altså

kategorisere hver pixel i en 3-dimensionel tensor, hvor de forskellige RGB-værdier forholder sig til en position i tensoren. Dette illustreres øverst på figur 2, hvis man ser på farverne i forhold til rød-, grøn- og blå-værdier.

Et farvet billede, der har rækker m og søjler n , er derfor en $(m \times n \times 3)$ -tensor, da dybdens størrelse afhænger af antallet af farver i en pixel. Hvis man derfor har et grayscale billede med størrelsen $256 \times 256 \times 1$, vil man kunne opbevare billedet i en 2-dimensionel matrix, $\mathbb{R}^{m \times n \times 1}$. Størrelsen af dette billede ville være $256 \times 256 \times 1 = 65.536$ bytes. Hvis man derimod har samme billede i farver, skal man opbevare billedet i en 3-dimensionel tensor, $\mathbb{R}^{m \times n \times 3}$. Størrelsen af dette billede vil være $256 \times 256 \times 3 = 196.608$ bytes. Farvede billeder kræver derfor større opbevaringskapacitet og længere tid for computeren at behandle, hvilket er vigtigt i machine

Figur 2: Dawani, Jay, 2020, *Hands-on mathematics for deep learning*, link:<https://www.yumpu.com/xx/document/read/65487454/hands-on-mathematics-for-deep-learning-by-jay>



⁷Wu, Jianxin, Introduction to Convolutional Neural Networks

learning⁸.

Convolution-operation

Der findes forskellige neural networks i machine learning, hvor convolutional neural network er en af de mest kendte og vigtigste. I et convolutional neural network, er convolution essentielt, hvilket er årsagen til modellens navn. I matematik skrives en convolution-operation således:

$$(f * g)(x) = \int f(t)g(t - x)dt$$

Her er input en funktion f og en funktion g , som er en kernel. Når man convolverer disse to funktioner, får man et output, som er en ny matrix. Hvert element udregnes fra en given matrix f og en kernel-matrix g . I machine learning bruger man diskret convolution, som skrives:

$$(f * g)(x) = \sum_t f(t)g(t - x)dt$$

Hvis man f.eks har en 3×3 -tensor og en 2×2 -kernel, kan man beskrive output, på følgende måde:

$$\begin{bmatrix} I_{1,1} & I_{1,2} & I_{1,3} \\ I_{2,1} & I_{2,2} & I_{2,3} \\ I_{3,1} & I_{3,2} & I_{3,3} \end{bmatrix} * \begin{bmatrix} K_{1,1} & K_{1,2} \\ K_{2,1} & K_{2,2} \end{bmatrix} = \begin{bmatrix} O_{1,1} & O_{1,2} \\ O_{2,1} & O_{2,2} \end{bmatrix}$$

Hvor man beregne O_{ij} , med følgende formel:

$$O_{ij} = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i - m, j - n)$$

Convolution illustreres ved, at funktionen g ”scanner” over givne input f og udregner elementerne i den nye matrix O , med overstående formel. Convolutionen starter i øverste-venstre hjørne, hvor den herefter går en søjle til højre. Når convolutionen har nået den yderste søjle, går den over til den venstre søjle og en række ned. Dette gør den, indtil alle elementerne er beregnet⁹.

Et eksempel på en to-dimensionel convolution-operation er i *bilag 1* med en 3×4 -matrix f og en 2×2 -kernel g .

⁸Dawani, Jay, 2020, Hands-on mathematics for deep learning

⁹Dawani, Jay, 2020, *Hands-on mathematics for deep learning*

$$\begin{bmatrix} 2 & 3 & 6 & 9 \\ 1 & 9 & 5 & 1 \\ 4 & 7 & 8 & 5 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} = \sum_{m=1}^3 \sum_{n=1}^4 I(m,n)K(i-m,j-n) \quad (1)$$

$$O_{1,1} = 2 \cdot 1 + 3 \cdot 2 + 1 \cdot 3 + 9 \cdot 1 = 20 \quad (2)$$

$$O_{1,2} = 3 \cdot 1 + 6 \cdot 2 + 9 \cdot 3 + 5 \cdot 1 = 47 \quad (3)$$

$$O_{1,3} = 6 \cdot 1 + 9 \cdot 2 + 5 \cdot 3 + 1 \cdot 1 = 40 \quad (4)$$

$$O_{2,1} = 1 \cdot 1 + 9 \cdot 2 + 4 \cdot 3 + 7 \cdot 1 = 38 \quad (5)$$

$$O_{2,2} = 9 \cdot 1 + 5 \cdot 2 + 7 \cdot 3 + 8 \cdot 1 = 48 \quad (6)$$

$$O_{2,3} = 5 \cdot 1 + 1 \cdot 2 + 8 \cdot 3 + 5 \cdot 1 = 36 \quad (7)$$

$$\begin{bmatrix} 2 & 3 & 6 & 9 \\ 1 & 9 & 5 & 1 \\ 4 & 7 & 8 & 5 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 20 & 47 & 40 \\ 38 & 48 & 36 \end{bmatrix} \quad (8)$$

opgaven på *bilag 1* = $\begin{bmatrix} 20 & 47 & 40 \\ 38 & 48 & 36 \end{bmatrix}$.

Implementering af convolution-operation i *Python*

Da der i det ovenstående er redegjort for den matematiske del af *convolutional-operation*, vil jeg i det efterfølgende redegøre for, hvordan man reelt implementerer convolution-operation i machine learning via. kodnings-sproget *Python*. Nedenfor ses et program, som tager en givet 2-dimensionel matrix, som er inputet f , og convolutionere med en givet 2-dimensionel kernel g . Dette program virker kun på 2-dimensionelle matricer, hvilket der tages højde for i denne redegørelse.

```

1 import numpy as np
2 def convolution(I,k):
3     assert(I.ndim == 2)
4     assert(k.ndim == 2)
5
6     I_x = I.shape[1]
7     I_y = I.shape[0]
8     k_x = k.shape[1]
9     k_y = k.shape[0]
```

```

10
11     Output_x = I_x-k_x+1
12     Output_y = I_y-k_y+1
13
14     Output = np.zeros((Output_y, Output_x))
15
16     for m in range(Output_y):
17         for n in range(Output_x):
18
19             for i in range(k_y):
20                 for j in range(k_x):
21                     if (m+i >= 0) and (m+i < I_y) and (n+j >= 0) and (n+j <
I_x):
22                         Output[m,n] = Output[m,n] + k[i,j]*I[i+m,j+n]
23
24
25 if __name__=="__main__":
26     Input_data = np.array([[2,3,6,9],
27                           [1,9,5,1],
28                           [4,7,8,5]])
29
30     kernel = np.array([[1,2],
31                       [3,1]])
32
33     print(convolution(Input_data, kernel))
34 >>> [[20, 47, 40],
35          [38, 48, 36]]

```

Jeg har brugt NumPy til ovenstående algoritme, som er et python-baseret bibliotek. NumPy er både hurtigt og understøtter multidimensionelle matricer, hvilket gør det muligt for mig at udforme dette program¹⁰.

De første linjer i programmet sikrer, at det givende input og kernel er to-dimensionelle:

```

3     assert(I.ndim == 2)
4     assert(k.ndim == 2)

```

Herefter definerer programmet x og y eller m og n dimensionerne til begge matricer. Her bruges `.shape[]`, til at finde dimensionerne til arrays og i dette tilfælde de to matricer. Her betegner 1, x/n-dimensionen og 0 y/m-dimensionen.

¹⁰NumPy.org, link:<https://numpy.org>

```

6     I_x = I.shape[1]
7     I_y = I.shape[0]
8     k_x = k.shape[1]
9     k_y = k.shape[0]
10
11    Output_x = I_x - k_x + 1
12    Output_y = I_y - k_y + 1

```

Ud fra inputet f og kernellen g 's dimensioner kan man udregne outputets dimensioner, som ses ved $\text{Output}_x = I_x - k_x + 1$ og $\text{Output}_y = I_y - k_y + 1$. Herefter laves en nulmatrix med dimensionerne beregnet fra tidligere. Dette gør det muligt blot at beregne elementerne i output-matrixen og hermed ændre elementerne i stedet for både at lave en ny matrix og udregne elementerne i output-matrixen i samme algoritme.

```
14     Output = np.zeros((Output_y, Output_x))
```

Når det er sikret, at dimensionerne til inputtet og kernellen er 2-dimensionelle og dimensionerne til output-matrixen er beregnet, kan algoritmen, som laver convolution, programmeres. Logikken bag denne algorithme kommer fra formlen fra tidligere.

$$O_{ij} = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

Man starter altså med at finde antallet af elementer i output-matrixen, hvilket gør det muligt at udregne, hvor mange gange kernellen skal ”scanne” inputtet. Dette ses i koden `for m in range(Output_y)` og `for n in range(Output_x):`. Herefter går kernellen over input-matrixen, hvor den nu kan udregne hvert element, hvis kravene `(m+i>=0) and (m+i < I_y) and (n+j >=0)`

`and (n+j < I_x)` gælder.

```

16     for m in range(Output_y):
17         for n in range(Output_x):
18
19             for i in range(k_y):
20                 for j in range(k_x):
21                     if (m+i >= 0) and (m+i < I_y) and (n+j >= 0) and (n+j < I_x):
22                         Output[m,n] = Output[m,n] + k[i,j]*I[i+m,j+n]
23
24     return Output

```

Disse krav sikrer sig, at kernellen ikke overskridt inputtets dimensioner, hvilket ville resultere i kernellen ville gå for langt i både x og y dimensionerne.

Herefter er inputtet og kernellen defineret og den definerede funktion *convolution* er brugt til at udregne convolutionen mellem inputtet og kernellen.

`if __name__ == "__main__":` er blot brugt for at give overblik over, hvornår programmet skal lave udregninger med defineret input og kernel.

```

25 if __name__=="__main__":
26     Input_data = np.array([[2,3,6,9],
27                           [1,9,5,1],
28                           [4,7,8,5]])
29     kernel = np.array([[1,2],
30                        [3,1]])
31     print(convolution(Input_data, kernel))

```

I det ovenstående har jeg redegjort for, hvordan man kan implementere convolution-operation i *Python*, jeg har illustreret det vha. opgaven i *bilag 1*. Dette bruges senere i opgaven til at forstå *convolutional neural networks*.

Neural network og convolutional neural network, *CNN*

I det følgende redegøres for, hvordan et neural network fungerer og hvordan et convolutional neural network adskiller sig fra et neural network.

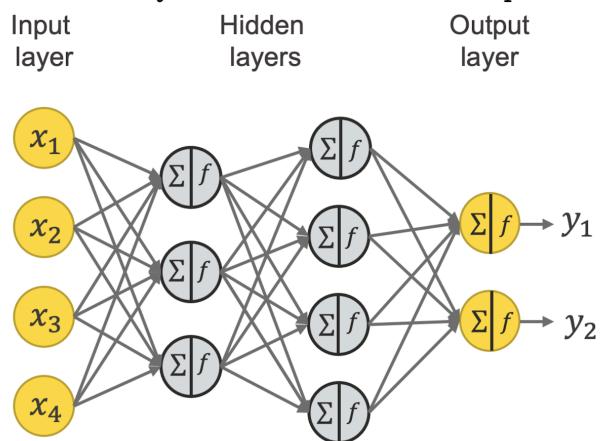
Neural network

For at forstå, hvordan et *convolutional neural network* adskiller sig fra et klassisk *neural network*, er det vigtigt først at forstå, hvad et klassisk *neural network* er og hvordan det fungerer. Et klassiskt *neural network* er baseret på hjer-

nens neuroner. Der er flere millioner neuroner i den menneskelige hjerne, som arbejder sammen og gør det muligt for mennesket f.eks at lære matematik eller genkende ansigter. Et klassiskt *neural network* er derfor inspireret af mennesket måde at lære på, hvilket er årsagen til navnet *neural network*.

Der findes forskellige modeller af neural networks, dog vil jeg redegøre for det mest almindelige klassiske/traditionelle neural network, som ses på figur 3. Denne model tager et givet input x og ganger herefter inputtet med en vægt w , som er specifik for det enkelte neuron. Hver streg, man ser på figur 3, har altså en givet vægt

Figur 3: Klassisk neural network, uden vægte. Melcher.K, *A Friendly Introduction to [Deep] Neural Networks*, link:<https://www.knime.com/blog/a-friendly-introduction-to-deep-neural-networks>



w . Hvis summen af de vægtede inputs er større end eller lig med grænseværdien for neuronet, så er outputtet 1. Hvis summen er mindre end grænseværdien for neuronet er outputtet 0. Denne beregning illustreres på figur 3 ved neuronet med sigma symbolet og aktiverings-funktionen f . Der findes forskellige aktiverings funktioner, som bruges i neural network. Den mest almindelige aktiveringsfunktion i et klassisk neural network er funktionen, som ses herunder:

$$y = \phi\left(\sum_i x_i w_i + b\right), \quad \phi = \frac{1}{1 + e^{\sum_{i=1}^n x_i w_i + b}}$$

Denne funktion kaldes sigmoid funktionen og er blot en af de mange aktiverings-funktioner neurale network bruger til at finde sammenhæng mellem input og output. Når man træner et neural network, giver man modellen et input og et resultat til det givende input. Herefter trænes neural network ved at ændre på de individuelle vægte w til de individuelle neuroner x og ændrer på b , som er bias. Bias b er en værdi, som tilføjes til inputtet ganget med vægten, hvilket gør det nemmere for det neurale network at finde det vigtigste input i aktiverings-funktionen¹¹. Et normalt program virker ved, at man giver bestemte regler til et program, så giver programmet nogle input og derefter får man et resultat. Et neural network får et input og resultaterne til outputtet og skal selv finde, justere og optimere disse regler, hvorefter det vil give et kvalificeret gæt på nyt input.

Convolutional neural network, CNN

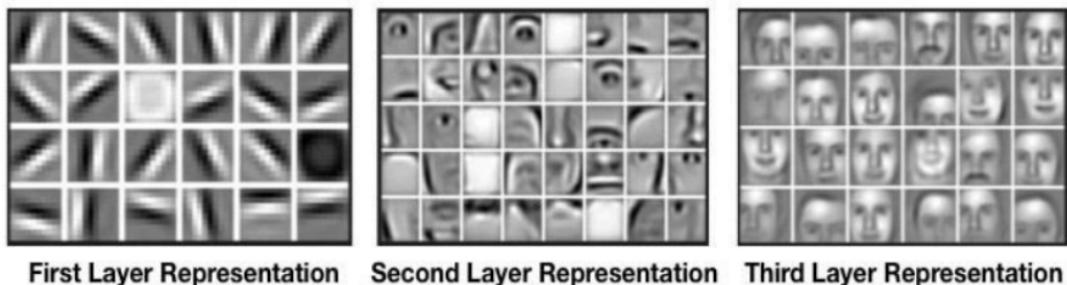
Convolutional neural networks er et af de mest populære og anerkendte neural networks. Dette fordi convolutional neural networks er gode til f.eks. at genkende ansigter, sprog og træne selvkørende biler. Da der tidligere er redegjort for matricer, tensors, convolution og klassiske neural network, kan jeg ud fra dette forklare, hvordan convolutional neural networks virker og hvordan det adskiller sig fra klassiske neural networks. Den største og mest relevante forskel er de to modellers arkitektur. På figur 3 ser man, hvordan de forskellige lag hænger sammen. I et convolutional neural network er størstedelen af disse klassifikationslag erstattet med convolutionslag. Disse convolutionslag laver først convolution mellem input og kernel, hvor laget herefter kan lære og fremhæve bestemte træk i f.eks. billeder. Der kræves flere convolutionslag for at genkende alle bestemte træk i et billede, da et convolutionslag kun ville fremhæve f.eks. vertikale træk i et billede¹².

¹¹Dawani, Jay, 2020, *Hands-on mathematics for deep learning*

¹²Dawani, Jay, 2020, *Hands-on mathematics for deep learning*

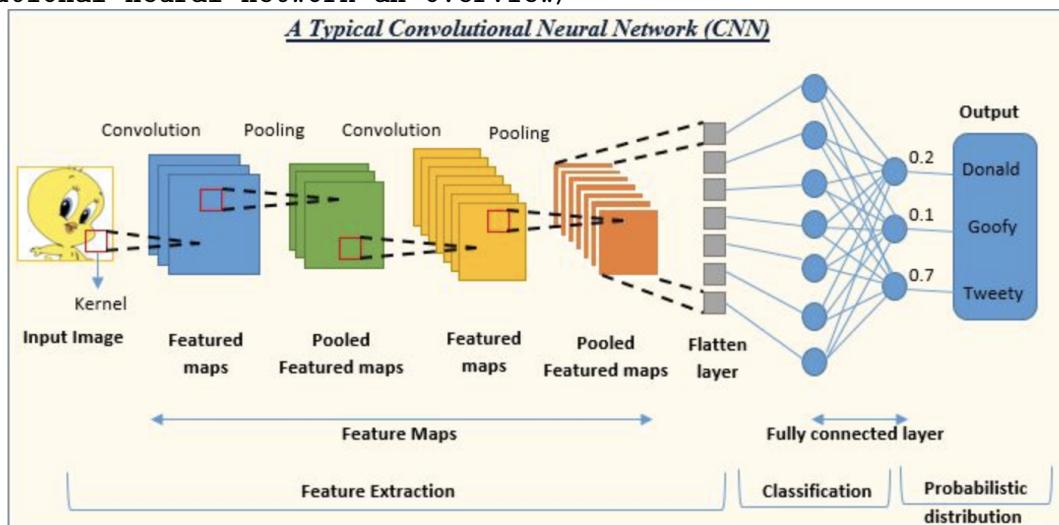
På figur 4 kan man se, hvordan første convolutionslag kun fremhæver linjer, hvor bestemte ansigtstræk bliver fremhævet mere efter flere convolutionslag.

Figur 4: Dawani, Jay, 2020, *Hands-on mathematics for deep learning*



På figur 5 kan man se arkitekturen for et convolutional neural network.

Figur 5: Arkitekturen af et convolutional neural network, Shah. S, *Convolutional Neural Network: An Overview*, link:<https://www.analyticsvidhya.com/blog/2022/01/convolutional-neural-network-an-overview/>



Årsagen til, at man bruger convolutional neural network fremfor klassiske neural network i f.eks. billedegenkendelse er mængden af faktorer, som påvirker outputtet. Hvis et klassisk neural network skal genkende et billede med dimensionerne 100×100 , vil inputstørrelsen være 10.000. Mængden af vægte w og bias b , som modellen skal forholde sig til, er derfor virkelig store. Hvis man derimod bruger et convolutional neural network, vil man efter convolutionslagene have mindre, men bedre input, som klassifikationslaget skal forholde sig til. Derfor ville et convolutional neural network være markant hurtigere og mere nøjagtig end et klassisk neural network. Klassifikationslaget er altså stadigvæk i et convolutional neural network, dog bliver inputtet først ændret af de forskellige convolutionslag, og så videreført til klassifikationslaget, som kan finde sammenhæng mellem input og output. I convolutional neural network er den typiske aktiverings-funktion ikke sigmoid funktionen, men Rectified Linear Unit(ReLU) funktionen.

Denne funktion bruger man, idet man opdagede, at ReLU funktionen gjorde træningsprocessen hurtigere, uden at gøre nøjagtigheden af modellen dårligere¹³.

$$f(x) = \begin{cases} 0, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$$

ReLU funktionen minder meget om en lineære funktion, hvilket virker godt i convolutional neural network. Derudover er funktionen nem at optimere, hvilket gør det nemmere at træne modellen¹⁴.

Efter hvert convolutionslag er gået igennem ReLU funktionen, er der et sidste ”pooling” lag. Et pooling-lag er ligesom en kernel og bruges oftest som en 2×2 -matrix. Dette lag ”scanner” matricerne og laver en ny matrix alt efter størrelsen af elementerne. I convolutional neural network bruger man Maxpooling, hvilket tager det største element i hver 2×2 -matrix og tilføjer elementet til en ny matrix. Dette gør det nemmere for klassifikationslaget at opfange vigtigt input, da klassifikationslaget skal forholde sig til mindre input. Et convolutional neural network bruges altså fremfor f.eks. klassisk neural network, når man har et stort input, hvor kun specifikke dele af inputtet har betydning.

”Hello world” i machine learning

I overstående er der redegjort for inputstypen af machine learning, hvad et neural network og convolutional neural network er og matematikken bag disse modeller. Derfor kan der nu laves et reelt convolutional neural network i programmeringsproget *Python*. Da dette vil være for svært selv for de bedste dataloger at lave et convolutional neural network fra bunden, bruger jeg Tensorflow. Tensorflow er et programmerings-bibliotek, som gør det nemmere at programmere et neural network.

Det første projekt, man møder, når man ønsker at lære om machine learning, er at lære et convolutional neural network at genkende cifferne fra 0 til 9. Jeg har vha. Tensorflow¹⁵ og medium.com¹⁶ programmeret og trænet et convolutional neural network til præcis dette. Jeg har opdelt dette projekt i 2 store dele, nemlig *model.py* og *GUI.py*. Selvom det nævnte projekt er interessant, er projektet for omfangsrigt at beskrive i min SRP.

¹³Dawani, Jay, 2020, *Hands-on mathematics for deep learning*

¹⁴Dawani, Jay, 2020, *Hands-on mathematics for deep learning*

¹⁵Tensorflow.com, *API*

¹⁶Quddoos, Talha, Medium.com *Get Started with Computer Vision by Building a Digit Recognition Model with Tensorflow*

Derfor vil jeg i nedenstående kun analysere *model.py*, da dette er det convolutional neural network.

```
1 import cv2
2 import keras
3 from keras.models import Sequential
4 from keras.layers.core import Dense, Dropout, Flatten
5 from keras.layers import Conv2D, MaxPool2D
6 from keras.datasets import mnist
7 from keras.utils import to_categorical
8
9 (X_train, y_train), (X_test, y_test) = mnist.load_data()
10
11 _, X_train_th = cv2.threshold(X_train, 127, 255, cv2.THRESH_BINARY)
12 _, X_test_th = cv2.threshold(X_test, 127, 255, cv2.THRESH_BINARY)
13
14 X_train = X_train_th.reshape(-1, 28, 28, 1)
15 X_test = X_test_th.reshape(-1, 28, 28, 1)
16
17 y_train = to_categorical(y_train, num_classes = 10)
18 y_test = to_categorical(y_test, num_classes = 10)
19
20 model = Sequential()
21 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape
22 =(28, 28, 1)))
23
24 model.add(Conv2D(64, (3, 3), activation='relu'))
25 model.add(MaxPool2D(pool_size=(2, 2)))
26
27 model.add(Dropout(0.25))
28 model.add(Flatten())
29 model.add(Dense(128, activation='relu'))
30
31 model.add(Dropout(0.5))
32 model.add(Dense(10, activation='softmax'))
33
34 model.compile(loss=keras.losses.categorical_crossentropy,
35                 optimizer=keras.optimizers.Adadelta(), metrics=['accuracy'])
36
37 history = model.fit(X_train, y_train, epochs=5, shuffle=True,
```

```

38         batch_size = 200, validation_data= (X_test, y_test))
39
40 model.save('model_data.h5')

```

Der bliver først importeret de forskellige biblioteker, som alle stammer fra Tensorflow¹⁷, bortset for cv2. OpenCV er et bibliotek, som gør computer vision nemmere¹⁸. OpenCV fylder ikke meget i selve modellen og er mere nødvendig i andre dele af projektet.

```

1 import cv2
2 import keras
3 from keras.models import Sequential
4 from keras.layers.core import Dense, Dropout, Flatten
5 from keras.layers import Conv2D, MaxPool2D
6 from keras.datasets import mnist
7 from keras.utils import to_categorical

```

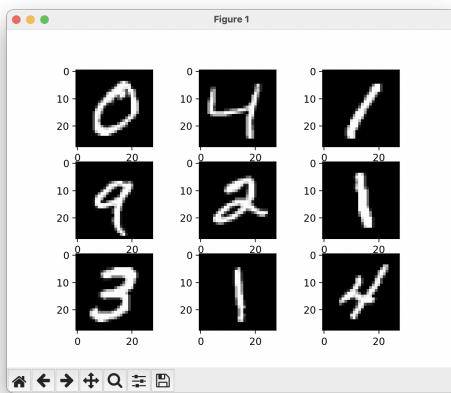
Herefter loader jeg datasættet `mnist`, som Tensorflow¹⁹ har sammensat, ind i mit projekt. Dette datasæt er specifikt til ciffer-genkendelse. Årsagen til, at datasættet er opdelt i `train` og `test`, er for at sammenligne de forskellige modellers resultater, i forhold til både hinanden og deres endelige resultater.

```

9 (X_train, y_train), (X_test, y_test) = mnist.load_data()

```

Her er billeder af de første 9 cifre i datasættet.



Herefter tager man dataen, som er grayscale, og giver alle de forskellige pixels en værdi og en farve mellem sort eller hvid, baseret efter den værdi de får givet. Fra tidligere ved vi, at grayscale-pixels har en værdi mellem 0 og 255. Hvis en pixel har en værdi mindre end eller lig med 127, bliver den sort og hvis den er større end eller lig med 128, bliver den hvid. Efter dette ændrer den dataens størrelse til 28×28 -matrix.

¹⁷Tensorflow.com

¹⁸Opencv.org

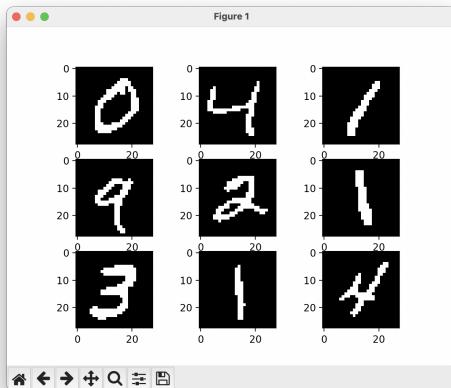
¹⁹[Tensorflow.com, link:<https://www.tensorflow.org/datasets/catalog/mnist>](https://www.tensorflow.org/datasets/catalog/mnist)

```

11 _,X_train_th = cv2.threshold(X_train,127,255,cv2.THRESH_BINARY)
12 _,X_test_th = cv2.threshold(X_test,127,255,cv2.THRESH_BINARY)
13
14 X_train = X_train_th.reshape(-1,28,28,1)
15 X_test = X_test_th.reshape(-1,28,28,1)

```

Dette illustreres herunder.



For at gøre det muligt for et convolutional neural network at forstå denne data, skal man ændre det til binær-matricer, da dette gør det muligt for algoritmen at forstå inputtet.

```

17 y_train = to_categorical(y_train, num_classes = 10)
18 y_test = to_categorical(y_test, num_classes = 10)

```

Da inputtet nu er gjort læsbart for algoritmen, er det nu muligt at modellere convolutional neural network til datasættet.

```

20 model = Sequential()
21 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape
                  =(28,28,1)))
22
23 model.add(Conv2D(64, (3, 3), activation='relu'))
24 model.add(MaxPool2D(pool_size=(2, 2)))
25
26 model.add(Dropout(0.25))
27 model.add(Flatten())
28 model.add(Dense(128, activation='relu'))
29
30 model.add(Dropout(0.5))
31 model.add(Dense(10, activation='softmax'))

```

`model = Sequential()` skaber selve modellen for netværket, hvorefter man tilføjer de convolutionelle lag ved `model.add(Conv2D(32, kernel_size(3,3), activation='relu', input_shape=(28,28,1)))`, `model.add(Conv2D(64,(3,3),activation='relu'))` og `model.add(MaxPool2D(pool_size=(2,2)))`. Herefter bruger man `model.add(Dropout(0.25))`, hvilket gør, at modellen ikke overanalyserer data. Dette gør `model.add(Dropout(0.25))` ved

at ignorere 25% af alle søger i inputtet. Hvis man ikke har `model.add(Dropout())`, vil bestemte data i modellen påvirke outputtet for meget. Derfor sikrer man, at alt data påvirker modellen lige meget, med denne kode. `model.add(Flatten())` gør input-matrixen til en 1-dimensionel matrix. Dette gør det muligt for klassifikationslaget at aflæse omregningerne. `model.add(Dense(128, activation='relu')` skaber klassifikationslaget og bruger aktiveringsfunktionen ReLU, som er redegjort for tidligere. Herefter bruges `model.add(Dropout(0.5))` igen, for at modellen ikke overanalyser nuværende data. Til sidst skaber man et sidste lag i klassifikationslaget med `model.add(Dense(10, activation='softmax'))`, hvor man anvender aktiverings-funktionen "softmax"²⁰. Efter man har lavet modellen, kan man starte træningen. Koden til dette ses under:

```

33 model.compile(loss=keras.losses.categorical_crossentropy,
34                 optimizer=keras.optimizers.Adadelta(), metrics=['accuracy'])
35 model.summary()
36
37 history = model.fit(X_train, y_train, epochs=5, shuffle=True,
38                       batch_size = 200, validation_data= (X_test, y_test))
39
40 model.save('model_data.h5')

```

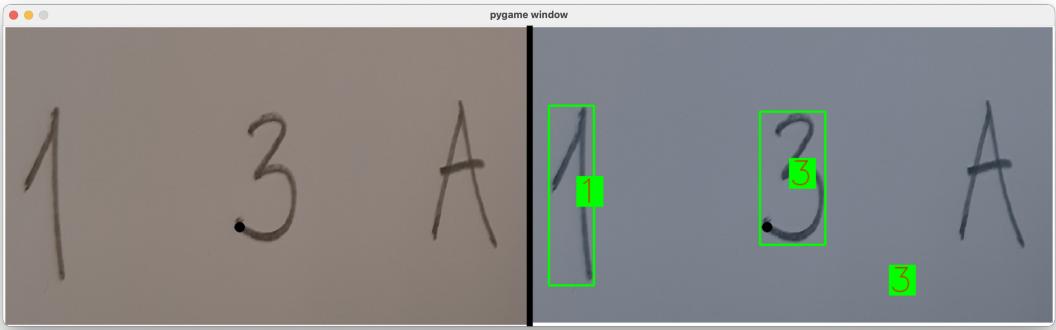
Her laver man en konfigurering af modellen med `model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizer.Adadelta(),metrics['accuracy'])`. Herefter får man en opsummering af det bedste lag og dimensionerne af inputtet i de individuelle lag vha. `model.summary()`. Til sidst træner man modellen med `model.fit(X_train, y_train, epochs = 5, shuffle=True, batch_size=200, validation_data=(X_test,y_test))`. Her giver man data, som modellen skal træne fra, antallet af træninger, om man vil ændre ordnen af data efter hver træning, mængden af data, der opdeles og til sidst sammenlignes med det generelle output. Sluttligt gemmes modellen med `model.save('model_data.h5')`, så jeg kan anvende modellen i de andre programmer²¹²².

Jeg har hermed vist, hvordan man programmerer modellen til et convolutional neural network. Jeg kan nu anvende bilag 2 på modellen og teste, om modellen virker. Her får jeg resultatet vist herunder:

²⁰Tensorflow.com, API

²¹Tensorflow.com, API

²²Quddoos, Talha, Medium.com, *Get Started with Computer Vision by Building a Digit Recognition Model with Tensorflow*



Her kan man se, at modellen kan genkende tallet 1 og 3, dog kan det ikke genkende bogstavet A. Dette viser, at modellen er nøjagtig og ikke bare antager, at man skriver et tal ved skrift. Dog tror modellen, at der er et lille 3 tal, hvor der egentlig ikke er noget. Modellen er derfor ikke helt perfekt, men kan genkende reelle cifre fra 0 til 9, derfor kan modellen vurderes vellykket.

Fordele og ulemper af kunstig intelligens

Jeg har nu redegjort for den matematiske og datalogiske del af et convolutional neural network, som er et mindre område af kunstig intelligens. Selvom mit projekt kan virke ”simpelt”, viser det, at man kan lære en computer at genkende billeder. Denne funktion har man udnyttet til f.eks. at finde tumorer hos brystkræft-patienter. Før i tiden tog det lægerne lang tid at nærværende patienters MR-scanninger og det var oftest svært at komme med konkrete svar. Man har derfor anvendt kunstig intelligens til at undersøge MR-scanninger af patienter med potentielle kræfttumorer. Her kan den kunstige intelligens lave et hurtigere og bedre svar end mange læger. Kunstig intelligens kan derfor hjælpe sundhedssektoren markant med at spare tid og give patienter hurtigere og mere præcise svar på måske livstruende sygedomme²³.

Noget som er rigtig interessant ved machine learning er, at modellerne selv bliver bedre, jo mere data man giver dem, hvilket er en teknologi, bilfirmaet Tesla bruger²⁴. Hvis man har en Tesla i USA, hvor autopilot er tilladt, bliver bilens autopilot bedre, jo mere man kører i bilen, da den kontinuerligt indsamler data fra kørslen. Selvom dette kan virke skræmmende for mange, da de føler deres data er privat, hjælper det dog samfundet generelt til en mere sikker og kørevenlig fremtid.

Der er også ulemper med denne essentielle dataindsamling til kunstig intelligens. Når f.eks Tesla indsamler data fra kørsel eller mobiltelefonen kræver face-id, er der samme problem, som man

²³Dileep. G og G Gianchandani Gyani. S, *Artificial Intelligence in Breast Cancer Screening and Diagnosis*

²⁴Tesla.com/AI, Neural network

har kæmpet med siden teknologi-boomet. Hacking. Ligeså vel som denne store mængde data er en fordel, kan den også vise sig at være en ulempe. Der kan ske store ulykker, hvis de forkerte mennesker hacker din bil, får adgang til dine bankkonti eller kan stjæle din identitet online. Der er også etiske problemstillinger, som vi skal forholde os til med kunstig intelligens. Hvis man f.eks. kører med sin bil i autopilot og man kører galt, hvem skal så tage skylden? I 2018 udgav MIT technologi review artiklen *Should a self-driving car kill the baby or the grandma? Depends on where you're from.* af Karen Hao.²⁵. Her blev to millioner mennesker spurgt om, hvordan de forholdt sig til ”trolley problemet”, hvor man enten skal ofre et ungt eller gammelt menneske. Dette er et reelt problem, da man nu både giver mennesker en værdi, men også fordi meningerne er blandede. Dette er blot en af de mange etiske problemstillinger, som man skal forholde sig til, når man anvender kunstig intelligens. Kunstig intelligens har derfor et kæmpe potentiale til at spare tid, penge og liv, men man skal også forholde sig kritisk til online-sikkerhed og personlige holdning, hvilket for mange kan virke for risikabelt.

Konklusion

Ud fra ovenstående redegørelser, analyse og diskussion kan jeg konkludere følgende:

En matrix, er et talskema, hvor tallene er organiseret i rækker i og søjler j . Der er forskellige regneregler for matricer såsom matrixmultiplikation og transponering. Ved transponering har jeg bevist formlen $(AB)^t = B^t A^t$, hvor t betegner transponering.

Jeg har generaliseret tidligere viden om matricer til 3. ordens tensor-matematik og vist, hvad convolution-operation er, hvordan det virker med en 2×2 kernel og hvorfor matricer er aktuelle i convolution-operation. Herpå har jeg bygget et *Python* program, som udregner convolution-operation med et givet input og kernel.

Jeg har både matematisk og vha. *python*-programmet udregnet regnestykket på bilag 1.

Med den viden, som jeg har fået fra ovenstående, har jeg redegjort for et convolutional neural network og et neural network. Efter disse redegørelser har jeg forklaret de markant forskelle mellem et neural network og et convolutional neural network. Disse forskelle var arkitekturen af de forskellige modeller og hvordan mængden af input var relevant.

Efter redegørelsen af et convolutional neural network har jeg programmeret et convolutional neural network i *Python* vha. bibliotekerne Tensorflow og OpenCv, som kan genkende cifrene fra 0 til 9.

²⁵MIT technologi review, Hao. Karen, *Should a self-driving car kill the baby or the grandma? Depends on where you're from.*

Jeg har analyseret koden bag modellen og anvendt programmet på bilag 2.

Jeg har diskuteret, hvorfor man med fordel kan anvende kunstig intelligens, som machine learning, i sundhedssektoren, da man ville kunne spare tid og give mere præcise svar med implementering af kunstig intelligens samt i bilindustrien.

Slutteligt kan jeg konkludere, at der både er fordele og ulemper ved brug kunstig intelligens industrien.

Fordele kan være forbedret kørekortført vha. autopilot, hurtigere og mere patientsikre svar på f.eks. MR-scanninger samt tidsbesparelser for sundhedspersonale f.eks. læger. Af ulemper kan nævnes risiko for brud på it-sikkerhed i form af hacking. Afgivelse af store mængder personlig data til firmaer, der kan misbruges. Derudover er der en del etiske dilemmaer, som bliver skabt, hvis man implementerer kunstig intelligens i f.eks. selvkørende biler. Så hvem vil få skylden, hvis de kørte galt med autopilot? Så kan man give en algoritme skylden, når kunstig intelligens er menneskeskabt?

Litteraturliste

Tekster:

Ridder Ebbesen, Grete, Matricer og lineære ligningssystemer, Sidst besøgt: d.27/3/2023
link: <https://www.uvmat.dk/paradig/not/n242-1.pdf>

Wikipedia, Transponering(matematik), besøgt d. 21/3/2023

link: [https://da.wikipedia.org/wiki/Transponering_\(matematik\)](https://da.wikipedia.org/wiki/Transponering_(matematik))

Wikipedia, Matrix, besøgt d. 21/3/2023

link: <https://da.wikipedia.org/wiki/Matrix>

University of Manchester, Prerequisite Material, Sidst besøgt: d. 27/3/2023

link: <https://personalpages.manchester.ac.uk/staff/Mark.Kambites/la/chap1.pdf>

Torfi, Amirsina

2020, *Practical linear algebra for machine learning*

link:<https://leanpub.com/linearalgebraML>

Dawani, Jay

2020, *Hands-on mathematics for deep learning*, Packt Publishing Ltd., Birmingham, UK, Sidst besøgt: 30/3/2023

link:<https://www.yumpu.com/xx/document/read/65487454/hands-on-mathematics-for-deep-learning>

Wu, Jianxin, Introduction to Convolutional Neural Networks

link: <https://cs.nju.edu.cn/wujx/paper/CNN.pdf>

Quddoos, Talha, Medium.com, *Get Started with Computer Vision by Building a Digit Recognition Model with Tensorflow*, Sidst besøgt: 27/3/2023

link: <https://medium.com/artificialis/get-started-with-computer-vision-by-building-a-digi>

Tensorflow.com, API, Sidst besøgt: 27/3/2023

link: https://www.tensorflow.org/api_docs/python/tf/keras/datasets/mnist/load_data

Opencv.org, Sidst besøgt: d. 27/3/2023 link: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

Tesla.com. Sidst besøg: d. 28/3/2023

link: <https://www.tesla.com/AI>

Dileep. G og G Gianchandani Gyani. S, National library of Medicin, *Artificial Intelligence in Breast Cancer Screening and Diagnosis* Sidst besøgt: 28/3/2023

link: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9650950/>

MIT technologi review, Hao. Karen, *Should a self-driving car kill the baby or the grandma? Depends on where you're from.*, Sidst besøgt: 28/3/2023

link: <https://www.technologyreview.com/2018/10/24/139313/a-global-ethics-study-aims-to-help>

Melcher.K, *A Friendly Introduction to [Deep] Neural Networks*, Sidst besøgt: 29/8/2023

link: <https://www.knime.com/blog/a-friendly-introduction-to-deep-neural-networks>

Shah. S, *Convolutional Neural Network: An Overview*, Sidst besøgt: 29/3/2023

link: <https://www.analyticsvidhya.com/blog/2022/01/convolutional-neural-network-an-overview/>

Stackexchange.com, *Simple visualization of 3D matrix*, Sidst besøgt: 29/3/2023

link: <https://tex.stackexchange.com/questions/300109/simple-visualization-of-3d-matrix>

NumPy.org, Sidst besøgt: 29/3/2023

link: <https://numpy.org>

Bilag

Bilag 1

● ● SCT. KNUDS GYMNASIUM

Bilag 1

Udfør udregningen nedenfor, hvor * er notationen for convolution-operatoren

$$\begin{bmatrix} 2 & 3 & 6 & 9 \\ 1 & 9 & 5 & 1 \\ 4 & 7 & 8 & 5 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix}$$



Bilag 2

