HAN_UNIVERSITY
OF APPLIED SCIENCES

# Deep Learning report

JULIAN VAN ZWOL, 2154913, JJP.VANZWOL@STUDENT.HAN.NL
SOHRAB HAKIMI, 2177196, S.HAKIMI1@STUDENT.HAN.NL
ROEL VAN EETEN, 2107391, R.VANEETEN@STUDENT.HAN.NL

MINOR: EVM – EVD3
GROUP: D-H-EVML
DATE: 06-01-2026

*By submitting this portfolio the authors certify that this is their original work, and they have cited all the referenced materials properly.*

# Contents

# 1 INTRODUCTION

Dit document introduceert ons Deep Learning (DL) portfolio dat wij ontwikkelen binnen de minor Embedded Vision & Machine Learning. Als projectgroep werken wij gezamenlijk aan het ontwerpen, trainen en evalueren van een deep learning model, met als casus het herkennen van schilderijen van Piet Mondriaan. Dit portfolio vormt een belangrijk onderdeel binnen de minor, waarin we ervaring opdoen met het toepassen van machine learning en vision-technieken in een complete workflow.

Deep learning speelt een steeds grotere rol binnen beeldanalyse en pattern recognition. Vooral Convolutional Neural Networks (CNN's) zijn zeer geschikt voor het automatisch herkennen van visuele patronen. Deze technieken sluiten goed aan bij onze interessegebieden, waarin visuele systemen en machine learning worden gebruikt om objecten, stijlen en scènes automatisch te classificeren. In dit project onderzoeken we hoe een CNN drie Mondriaan-schilderijen kan herkennen, en hoe het model onderscheid kan maken tussen Mondriaan- en niet-Mondriaan-schilderijen. De gestileerde kenmerken van *De Stijl* maken deze opdracht een interessant domein om de kracht en beperkingen van CNN's te analyseren.

Het project sluit nauw aan bij de onderwerpen die in de minor worden behandeld, zoals computer vision, datastructuren voor AI en embedded toepassingen. Daarnaast vormt dit portfolio een directe voorbereiding op ons main project binnen de minor, waar we een vergelijkbare visuele herkenningsopdracht uitvoeren. Dit DL-prototype helpt ons te onderzoeken of deep learning geschikt is voor het herkennen van Mondriaan-schilderijen, en geeft waardevolle inzichten voor het toepassen van deze kennis in het hoofdproject.

Aan het einde van dit project willen wij:

- Begrijpen hoe CNN-architecturen zijn opgebouwd en functioneren.
- Een dataset kunnen samenstellen, uitbreiden en voorbereiden voor training.
- Een deep learning model kunnen trainen, valideren en evalueren.
- Verbeteringen kunnen doorvoeren op basis van performance-analyse.
- De toepasbaarheid van DL binnen ons main project kunnen onderbouwen.

# 2 PROBLEM STATEMENT

Het probleem omgaande dit project, is dat als museum gast je door een zaal loopt waarbij verschillende Mondriaan schilderijen hangen. Bij deze schilderijen wordt niet altijd weergegeven welke variant van Piet Mondriaan dit is en wat de geschiedenis van dit schilderij is. Onze doelstelling is hier een oplossing voor te verzinnen. Deze oplossing moet minimaal worden voorzien van een deep learning algoritme. De verdere eisen aan dit project zijn hieronder beschreven in functioneel en niet functionele specificaties. De specificaties zijn beschreven aan de hand van de prioriteitsbeschrijving Moscow(Must, Should, Could of Would).

## 2.1 Functionele specificaties

Functionele requirements beschrijven wat het systeem moet doen, dus de specifieke taken, functies en interacties die het programma moet uitvoeren om de gewenste resultaten te leveren. De eisen in dit hoofdstuk zijn verzameld uit gesprekken met stakeholders, zodat het systeem aansluit bij de behoeften en verwachtingen van de gebruikers.

| Id | Prio | Omschrijving |
|---|---|---|
| F1 | M | Het systeem moet de vier vooraf gedefinieerde schilderijen (Mondriaan 1, Mondriaan 2, Mondriaan 3 en Mondriaan 4) correct herkennen met een nauwkeurigheid van minimaal 60%. |
| F2 | M | Na inladen van de afbelding moet het systeem binnen 4 seconden de titel van het schilderij zichtbaar tonen op het scherm van de gebruiker. |
| F3 | M | Wanneer een schilderij niet overeenkomt met een van de vier vooraf gedefinieerde Mondriaan-schilderijen, moet het systeem binnen 4 seconden het schilderij classificeren als zijnde niet Mondriaan. |
| F4 | M | Het systeem functioneert betrouwbaar bij kantoorverlichting. |
| F5 | M | Het systeem moet correcte herkenning uitvoeren bij kijkhoeken tot maximaal 20 graden ten opzichte van het schilderijoppervlak. |
| F6 | M | Het systeem moet betrouwbare herkenning uitvoeren bij een afstand tussen 30 cm en 50 cm van het schilderij. |

## 2.2 Non functionele specificaties

Non-functionele requirements specificeren de kwaliteits- en prestatiecriteria waaraan het systeem moet voldoen, zoals betrouwbaarheid, nauwkeurigheid en verwerkingsefficiëntie. De meeste eisen zijn afgeleid uit gesprekken met stakeholders, terwijl eisen gemarkeerd met * voortkomen uit of zijn verfijnd op basis van onderzoeksvragen en dus niet direct uit stakeholderinteracties.

| Id | Prio | Omschrijving |
|---|---|---|
| NF1 | M | Het systeem moet een afbeelding kunnen inladen van JPEG-formaat (.jpg). |
| NF2 | M | Het systeem moet alle ingevoerde afbeeldingen automatisch schalen naar een resolutie van 1920×1080 pixels (Full HD) vóór verdere verwerking, om een consistente beeldkwaliteit en verwerkingssnelheid te garanderen. |
| NF3 | M | Het getrainde model moet bij validatie met een onafhankelijke testdataset een herkenningsnauwkeurigheid van minimaal 80% behalen voor de vier vooraf gedefinieerde schilderijen. |
| NF4 | M | De trainingsdataset moet bestaan uit minimaal 100 gelabelde afbeeldingen per schilderij, met variatie in hoek en afstand om realistische omstandigheden te simuleren. |
| NF5 | M | Het systeem moet gebruik maken van een CNN. |
| NF6 | C | Het systeem moet bij elke voorspelling een zekerheidsscore genereren tussen 0 en 100%, die de betrouwbaarheid van de classificatie aangeeft. |
| NF7 | C | Wanneer de zekerheidsscore van een voorspelling lager is dan 50%, moet het systeem automatisch de gebruiker instrueren om één of meerdere extra beelden te maken voor een nieuwe analyse. |

# 3 DATA AUGMENTATION AND PREPROCESSING

## 3.1 Vergroten van de dataset

Uit het onderzoek document (van Zwol, Hakimi, & van Eeten, 2025) blijkt dat de onderzoeksvragen gekeken wordt naar de praktische omstandigheden die de prestaties van het systeem beïnvloeden, zoals lichtvariaties, kijkhoek, afstand tot het schilderij, omlijsting en achtergrond. In musea kunnen factoren zoals licht, kijkhoek, afstand, reflecties en achtergrond sterk variëren. Voor dit project zijn deze variabelen als volgt gespecificeerd: lichtintensiteit varieert tussen 50 en 200 lux, de kijkhoek kan variëren van 0° tot 50°, en de afstand tot het schilderij ligt tussen 1 en 2 meter. Onderzoek heeft aangetoond dat het materiaal dat voor het schilderij hangt nauwelijks invloed heeft op de herkenning, terwijl de omlijsting en variatie in achtergrond in dit project buiten beschouwing worden gelaten.

Voor het trainen van een betrouwbaar deep learning-model is een uitgebreide dataset noodzakelijk die representatief is voor deze praktische variaties. Gezien het beperkte tijdsbestek is het niet haalbaar om handmatig voldoende afbeeldingen te verzamelen die al deze variaties bevatten. Om deze reden wordt dataaugmentatie toegepast, waarmee nieuwe afbeeldingen kunnen worden gecreëerd.

De geselecteerde augmentatietechnieken zijn:

- RandomRotation en RandomShear, die variaties in kijkhoek simuleren. Hoewel RandomShear het perspectief en de bijbehorende foreshortening niet volledig nabootst, benadert het de variatie in kijkhoek het dichtst.

```
# Simuleert kijkhoekvariatie
keras.layers.RandomShear(0.2),
keras.layers.RandomRotation(0.1),
```

*Figuur 1 random rotation en random shear*

- RandomZoom, dat variatie in afstand tot het schilderij imiteert.

```
# Simuleert afstand tot het schilderij
keras.layers.RandomZoom(0.2),
```

*Figuur 2 random zoom*

- RandomBrightness en RandomContrast, die de variatie in lichtintensiteit simuleren.

```
# Simuleert lux-variatie
keras.layers.RandomBrightness(0.3),
keras.layers.RandomContrast(0.3),
```

*Figuur 3 random brightness en random contrast*

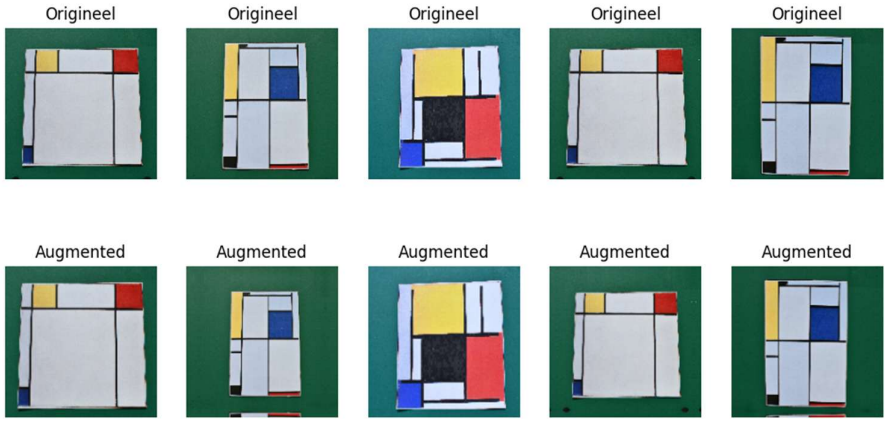
## 3.2 Augmentatie resultaten

Figuur 5 illustreert het effect van dataaugmentatie op basis van kijkhoekvariatie. In de bovenste rij worden de originele werken weergegeven, met hun kenmerkende rechte lijnen, primaire kleuren en rechthoekige vlakken. De onderste rij toont dezelfde composities na toepassing van de

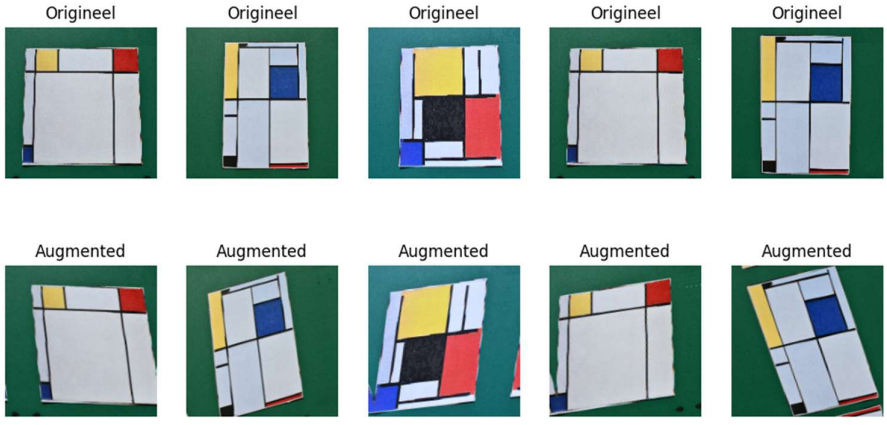
RandomShear- en RandomRotation-transformaties. Deze augmentaties zorgen voor perspectiefverschuivingen en rotaties, waardoor de oorspronkelijk rechte lijnen scheef komen te staan en er een driedimensionaal effect ontstaat.

Figuur 4 illustreert het effect van RandomZoom-augmentatie op dezelfde afbeeldingen. Door deze augmentatie worden delen van de composities ingezoomd, waardoor sommige elementen worden vergroot of afgesneden. Deze techniek simuleert variatie in kijkafstand en draagt bij aan de robuustheid van het model.

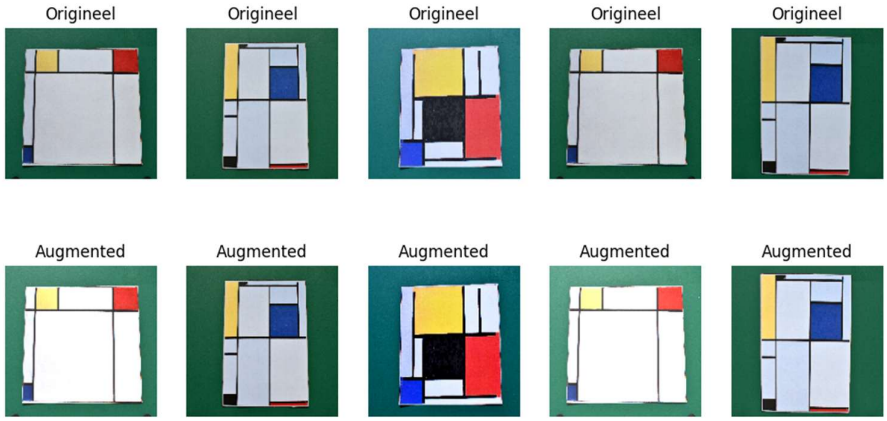
Ook Figuur 6 demonstreert het effect van RandomBrightness- en RandomContrast-augmentatie op dezelfde werken. Het effect is vooral zichtbaar in de witte en achtergrond vlakken, die lichter of donkerder kunnen zijn geworden, en in de primaire kleuren, waarvan de intensiteit varieert.




*Figuur 4 random zoom*

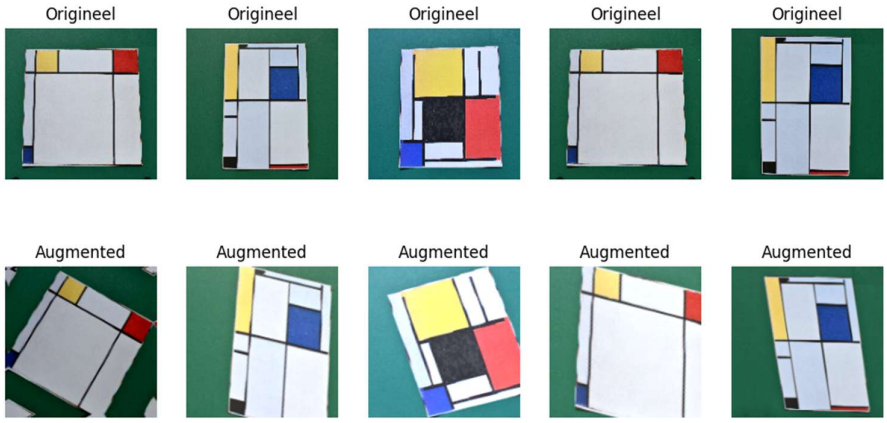



*Figuur 5 random rotation en random shear*




*Figuur 6 random brightness en random contrast*

Combineren we vervolgens alle augmentatie stappen tegelijk zien we de resultaten in Figuur 7.




*Figuur 7 gecombineerde features*

## 3.3 Voorbereiden van afbeeldingen voor CNN

Voor de verwerking van afbeeldingen door een convolutional neural network (CNN) is een gestructureerde preprocessing pipeline ontwikkeld die uit vier opeenvolgende stappen bestaat. Allereerst worden alle afbeeldingen geüniformeerd naar een resolutie van 244x244 pixels, wat enerzijds het laden in het geheugen mogelijk maakt en anderzijds voldoet aan de vaste inputvereisten van CNN-architecturen. Her resizen gebeurt in de utility functie image_dataset_from_directory zoals te zien in Figuur 8.

```python
image_size = (224, 224)
batch_size = 128

if __name__ == "__main__":
    train_ds, val_ds = keras.utils.image_dataset_from_directory(
        r"C:\workspace\evml\EVD3\mondriaan-detector-dl\data\fullset",
        validation_split=0.2,
        subset="both",
        seed=1337,
        image_size=image_size,
        batch_size=batch_size,
    )
```

*Figuur 8 inladen afbeeldingen*

Vervolgens wordt data augmentatie toegepast voorafgaand aan de normalisatie van pixelwaarden, met als doel de variatie binnen de dataset te vergroten, overfitting tegen te gaan en de robuustheid van het model te verbeteren.

In de derde stap worden de pixelwaarden geschaald van het oorspronkelijke bereik [0, 255] naar [0, 1], waardoor neurale netwerken efficiënter en stabieler leren doordat grote numerieke waarden die de training zouden kunnen domineren worden voorkomen.

Tot slot vindt normalisatie plaats waarbij de pixelwaarden per kleurkanaal worden gecentreerd rond een gemiddelde van nul met een standaarddeviatie van één, zodat alle features op hetzelfde schaalniveau opereren en het netwerk geen voorkeur ontwikkelt voor specifieke kanalen.

Deze drie stappen data augmentatie, rescaling en normalisatie worden binnen dit project geïntegreerd als lagen in de architectuur van het CNN-model zelf zoals te zien op Figuur 9. Hierdoor worden de preprocessing-transformaties dynamisch toegepast op de inputafbeeldingen, in plaats van dat ze vooraf op de dataset worden uitgevoerd.

```python
model = keras.Sequential([
    keras.Input(shape=input_shape),
    data_augmentation_layer,
    keras.layers.Rescaling(1./255),
    keras.layers.Normalization(axis=-1),
```

*Figuur 9 toepassing data augmentatie*

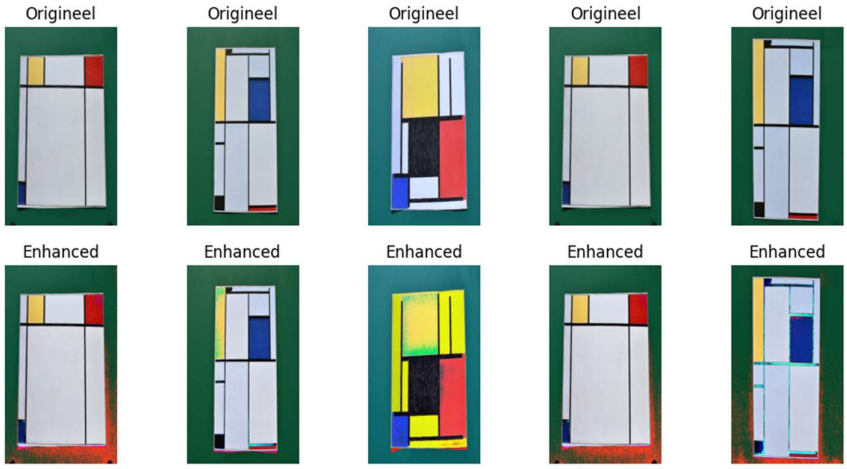
Deze preprocessing pipeline draagt bij aan het verduidelijken van relevante visuele patronen voor het CNN en vermindert de complexiteit door consistente en gestandaardiseerde input te leveren.

Daarnaast is onderzocht in hoeverre image enhancement de prestaties van het systeem kan verbeteren. Door het contrast en de helderheid van de afbeeldingen aan te passen voordat deze de convolutionele lagen van het netwerk bereiken, wordt beoogd het model robuuster te maken tegen variaties in belichting en visuele kenmerken. De implementatie van deze aanpassingen is als volgt vormgegeven:

```python
def enhance_images():
    return keras.Sequential([
        keras.layers.Lambda(lambda x: tf.image.adjust_contrast(x, 1.15)),
        keras.layers.Lambda(lambda x: tf.image.adjust_brightness(x, 0.05))
    ])
```

*Figuur 10 implementatie enhance images*

Echter leverde deze aanpak geen visuele verbetering op, zoals geïllustreerd in de betreffende Figuur 11.




*Figuur 11 enhanced images*

# 4 CNN ARCHITECTURE, TRAINING AND VALIDATION

## 4.1 CNN-architectuur

Voor het herkennen van Mondriaan-schilderijen is gekozen voor een zelfontworpen CNN met drie convolutionele lagen, gevolgd door pooling, dropout en fully connected lagen.

Voor het aantal filters is er gekozen voor in de eerste laag 24, waarna dit in de diepere lagen oploopt naar 48 en 64. Deze opbouw sluit aan bij hoe CNN's normaal werken: de eerste lagen herkennen eenvoudige kenmerken zoals lijnen en kleurvlakken, terwijl latere lagen complexere patronen leren.

Mondriaan-schilderijen bestaan uit grote, egale kleurvlakken en duidelijke rechte lijnen. Om deze structuren beter te herkennen is gekozen voor een kernel size van 7×7 in plaats van de gebruikelijke 3×3. Met kleinere kernels leken de feature maps sterk op het originele beeld, terwijl grotere kernels al vroeg zorgen voor meer abstractie en beter passende kenmerken.

Omdat de schilderijen vrij eenvoudig en geometrisch zijn opgebouwd, is een diep netwerk met veel lagen niet nodig. Drie convolutionele lagen zijn voldoende om de belangrijkste kenmerken te leren, zoals kleurverdeling, lijnen en de globale compositie van het schilderij.

Zonder extra maatregelen vertoonde het model de neiging om zich te richten op specifieke pixelpatronen in de trainingsdata, wat leidt tot overfitting. Het model presteert dan goed op bekende voorbeelden, maar slecht op nieuwe schilderijen. Om dit te voorkomen is een dropout-laag met een waarde van 0.4 toegevoegd vóór de dense laag. Tijdens training wordt hierdoor steeds de helft van de neuronen tijdelijk uitgeschakeld, waardoor het netwerk wordt gedwongen om robuustere en algemenere kenmerken te leren.

## 4.2 Transfer learning

Naast het eerder ontwikkelde Convolutional Neural Network (CNN) is ook gebruikgemaakt van transfer learning. Transfer learning is een techniek binnen deep learning waarbij een bestaand, voorgetraind model wordt hergebruikt voor een nieuwe, maar gerelateerde taak. Het voordeel hiervan is dat het model al algemene kenmerken heeft geleerd, zoals randen, vormen en patronen, waardoor minder trainingsdata nodig is en het leerproces sneller verloopt.

In dit project is gekozen voor MobileNetV2, een lichtgewicht en veelgebruikt neuraal netwerk dat vooraf is getraind op de ImageNet-dataset. Dit model is geschikt voor beeldclassificatie en wordt vaak toegepast wanneer rekencapaciteit beperkt is.

## 4.3 Optimalisatie

Voor het afstellen van de hyperparameters is gebruikgemaakt van Keras Tuner met de Hyperband-methode. De tuner maakt eerst veel verschillende varianten van het model en traint deze slechts kort, bijvoorbeeld één of twee epochs. Modellen die slecht presteren worden snel afgeschreven, terwijl de best presterende helft extra training krijgt. Dit proces herhaalt zich meerdere keren, waarbij steeds meer rekentijd wordt besteed aan de meest veelbelovende configuraties. Uiteindelijk krijgt alleen het beste model de volledige training van 10 epochs.

De uitkomsten van deze hyperparameterzoektocht zijn te vinden in Bijlage A. Hieruit blijkt dat de beste configuratie een learning rate van 0.001 en een dropout rate van 0.4 gebruikt.

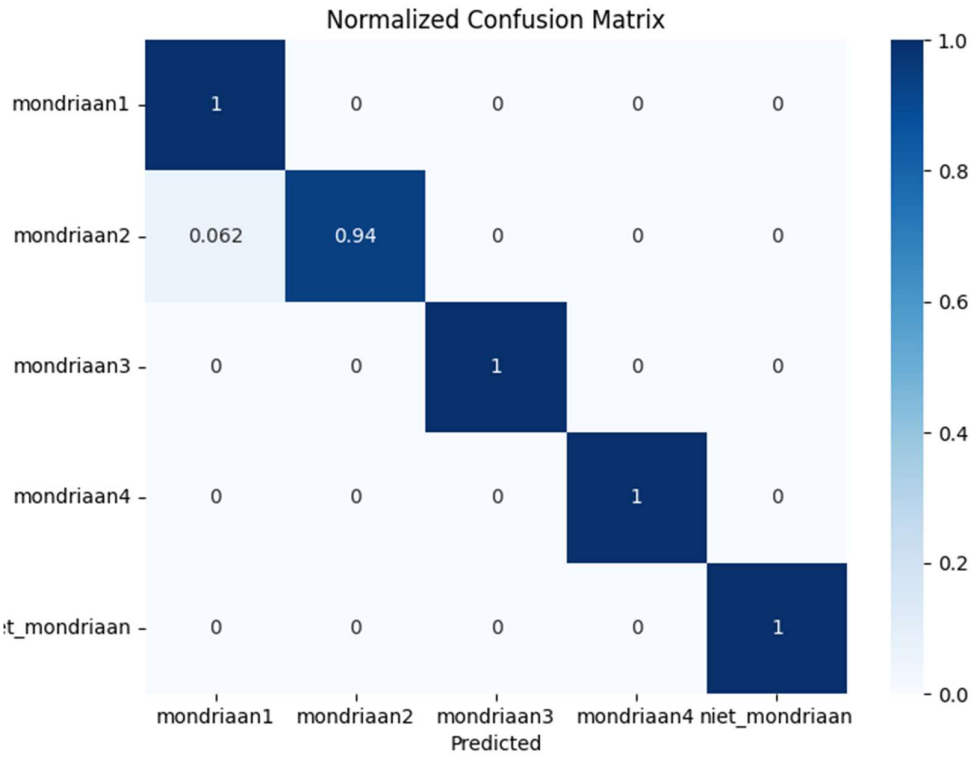
Als prestatiemaatstaf is gekozen voor *validation accuracy*. Dit is een duidelijke en praktische maat: het geeft simpelweg aan welk percentage van de afbeeldingen door het model correct wordt geclassificeerd.

## 4.4 Model evaluatie

Na het trainen van het model op de volledige training set, met de eerder bepaalde optimale hyperparameters, is de prestatie beoordeeld op een aparte testset. Deze testset is tijdens het trainen niet gebruikt, waardoor de evaluatie een eerlijk beeld geeft van hoe goed het model generaliseert naar nieuwe data.

De resultaten zien er als volgt uit:

- Accuracy: 0.9915966391563416
- F1-score: 0.9915

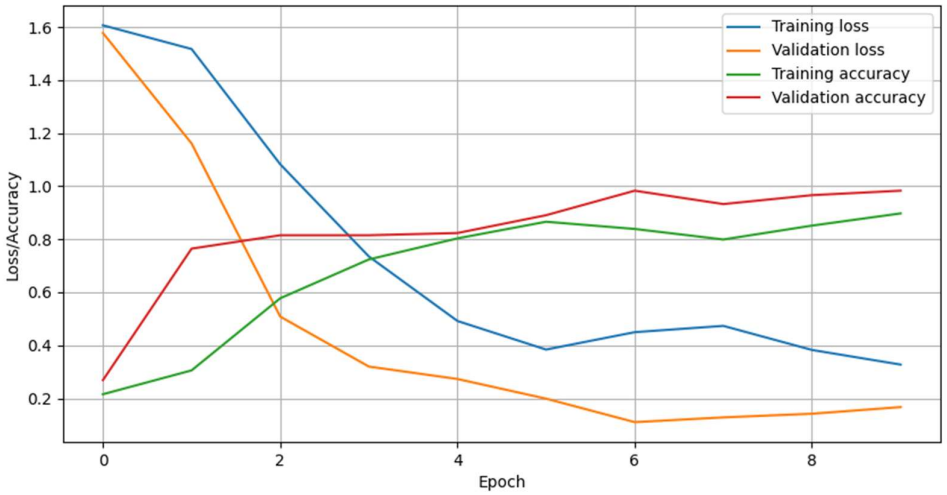



*Figuur 12 confusion matrix*

De confusion matrix laat zien dat het model vrijwel perfect presteert. Alle klassen worden grotendeels correct geclassificeerd, met waarden van 1.0 op de diagonaal voor *mondriaan1*, *mondriaan3*, *mondriaan4* en *niet Mondriaan*.

De enige noemenswaardige fout zit bij *mondriaan2*: ongeveer 6% van deze afbeeldingen wordt ten onrechte als *mondriaan1* geclassificeerd. Verder zijn er geen verwarringen met de *niet_mondriaan*-klasse, wat aangeeft dat het model goed onderscheid maakt tussen Mondriaan-schilderijen en andere afbeeldingen. Overall wijst de matrix op een zeer robuust en betrouwbaar model.

De learning curve ziet er als volgt uit:




*Figuur 13 learning curve*

Wat meteen opvalt, is dat de validation accuracy en validation loss beter zijn dan de trainingswaarden. Tijdens het trainen wordt het model bewust moeilijker gemaakt door het gebruik van dropout en data augmentation. Het krijgt vervormde afbeeldingen te zien en een deel van de neuronen wordt tijdelijk uitgeschakeld, waardoor de prestaties op de trainingsdata lager uitvallen.

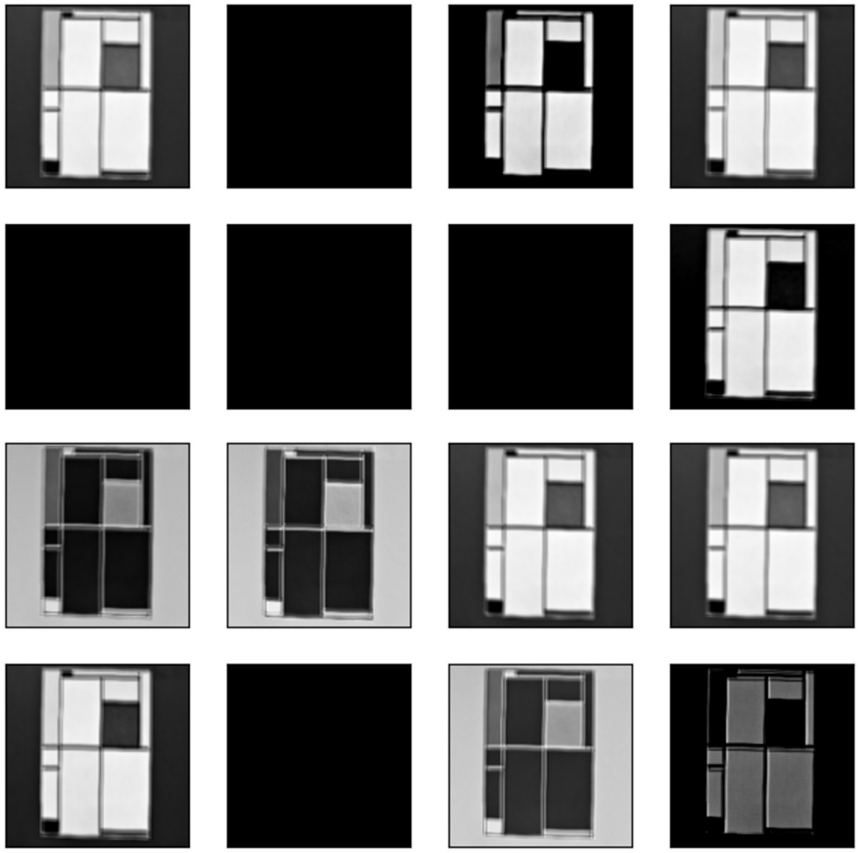
Bij de validatie gebeurt dit niet: alle neuronen staan aan en de afbeeldingen zijn onbewerkt. Hierdoor heeft het model het tijdens de validatie eenvoudiger, wat resulteert in betere scores. Dit laat zien dat de dropout-laag doet wat hij moet doen: het voorkomt dat het model simpelweg de trainingsdata uit het hoofd leert.

Data augmentation zorgt daarnaast voor veel variatie in de input. Doordat afbeeldingen bijvoorbeeld worden gesheared of ingezoomd, ziet het model nooit exact hetzelfde beeld twee keer. Hierdoor wordt het gedwongen om zich te richten op de echt relevante kenmerken, zoals lijnen, kleurvlakken en hun onderlinge verhoudingen, in plaats van op exacte pixelposities.

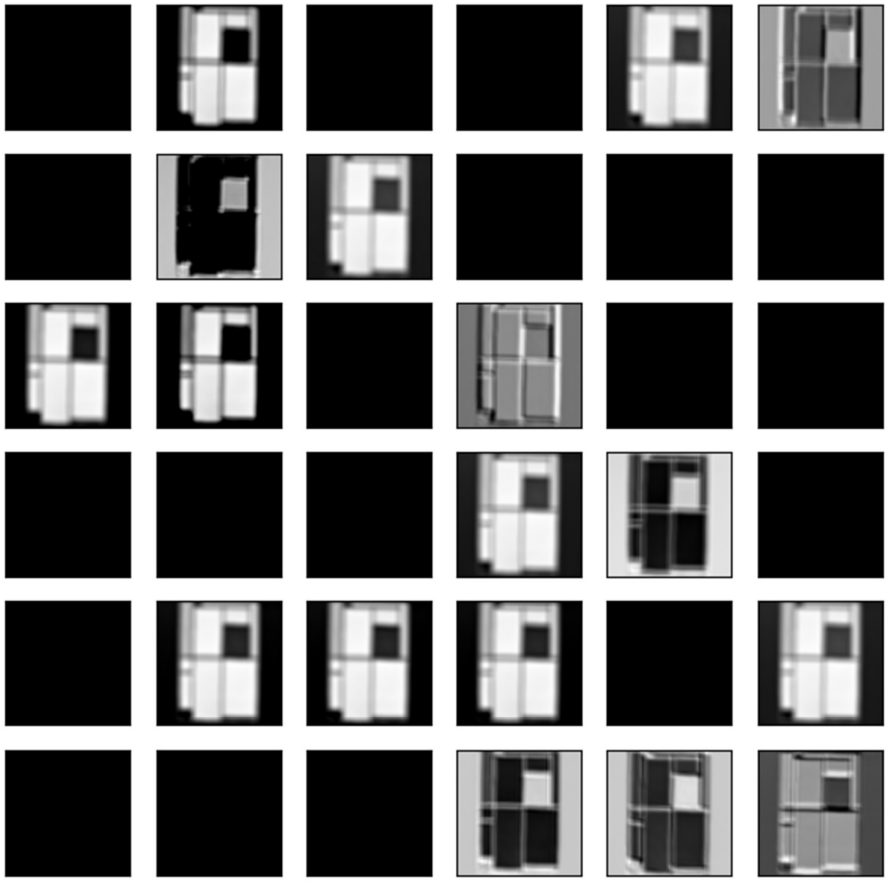
## 4.5 Visualisatie CNN learning

Om inzicht te krijgen in wat het CNN leert, zijn de activaties van verschillende lagen gevisualiseerd voor een voorbeeldinvoer.

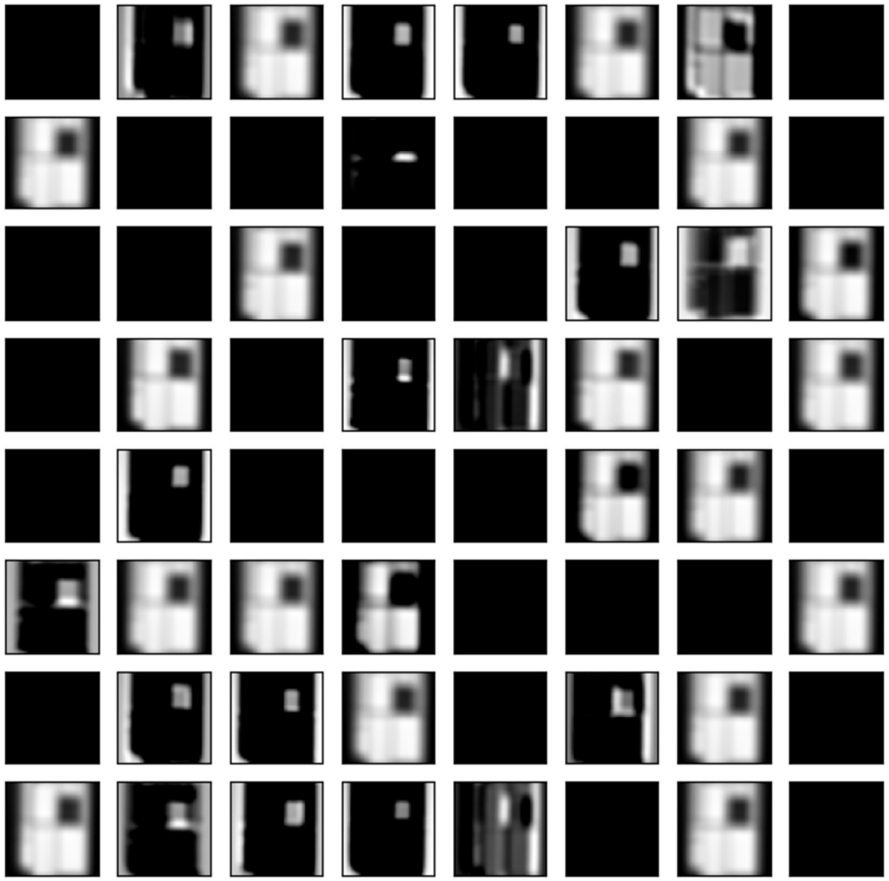
De eerste convolutionele laag detecteert low-level features. Denk hierbij aan de vlakken en lijnen. Deze features op een laag abstractieniveau zijn dus nog goed zichtbaar zoals te zien in Figuur 14. In de tweede laag gaan we naar een hoger abstractieniveau en worden de afbeeldingen waziger. Het model leert dus meer over de afbeelding maar in de visualisatie verlies je informatie. Dit zie je versterkt in de Figuur 16.



*Figuur 14 eerste laag*

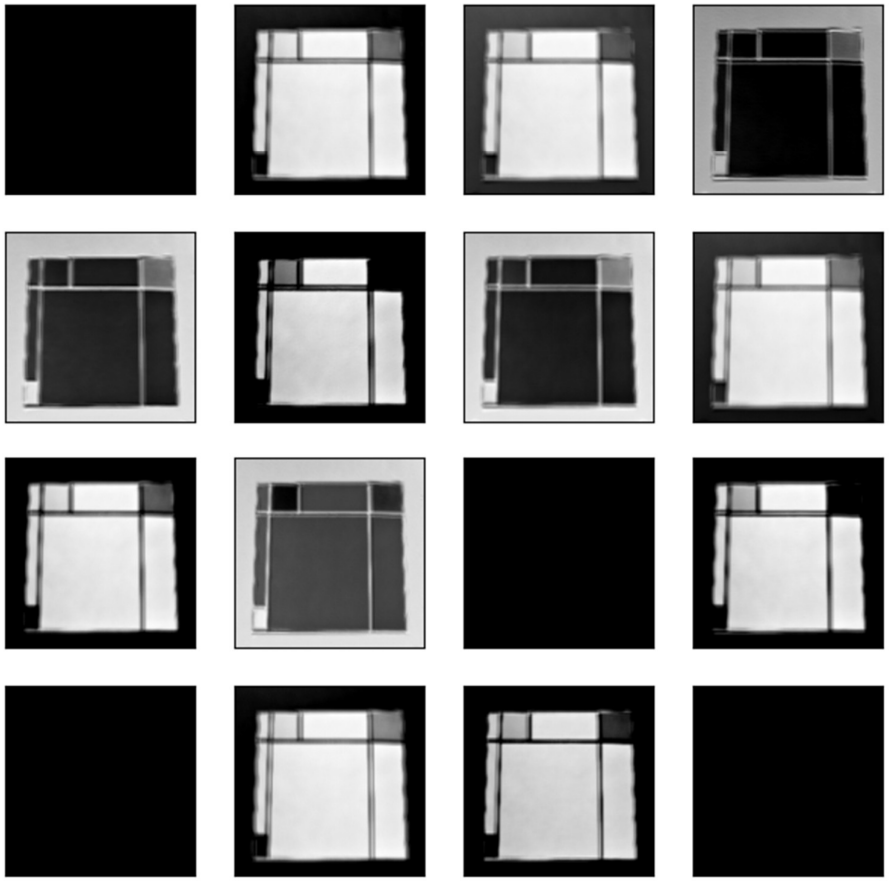


*Figuur 15 tweede laag*



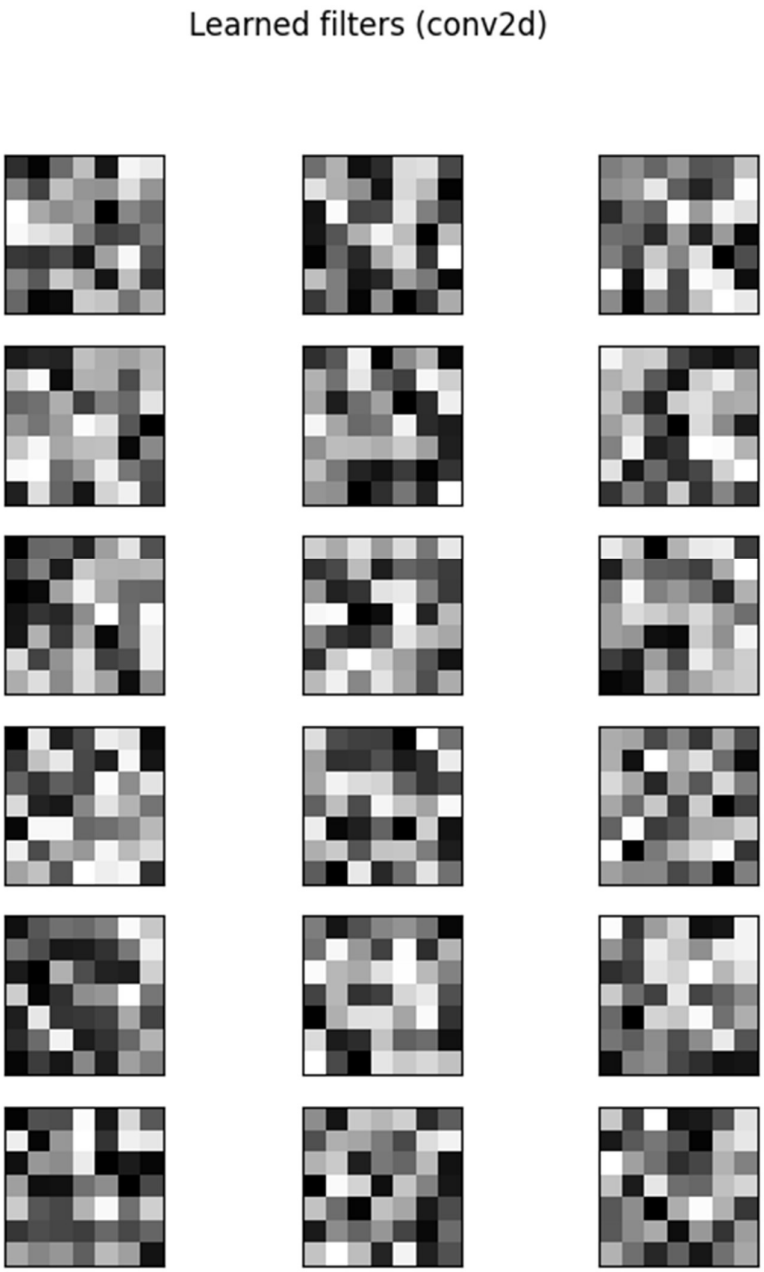
*Figuur 16 derde laag*

Ook zie je in iedere laag volledig zwarte visualisaties. Dit komt omdat niet iedere kernel aanslaat op iedere afbeelding. In Figuur 17 zie je dat kernels die niet aansloegen op de eerste afbeelding dat wel doen op de tweede afbeelding.



*Figuur 17 tweede afbeelding*

### 4.5.1 Filter visualisatie




*Figuur 18 learned filters*

## 4.6 Model MobileNetV2

Voor het toepassen van transfer learning is MobileNetV2 gebruikt als basisnetwerk. Dit basisnetwerk is uitgebreid met extra lagen die specifiek zijn afgestemd op het classificeren van Mondriaan en niet

monderiaan-schilderijen. Voorafgaand aan de classificatie worden de afbeeldingen eerst geaugmenteerd, zodat het model robuuster wordt voor variaties zoals rotatie, schaal en belichting.

In de eerste fase zijn de convolutionele lagen van MobileNetV2 bevroren. Dit betekent dat deze lagen niet worden aangepast tijdens het trainen. Alleen de nieuw toegevoegde dense-lagen aan het einde van het netwerk worden getraind.

Model: "functional_1"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer (InputLayer) | (None, 224, 224, 3) | 0 |
| sequential (Sequential) | (None, 224, 224, 3) | 0 |
| true_divide (TrueDivide) | (None, 224, 224, 3) | 0 |
| subtract (Subtract) | (None, 224, 224, 3) | 0 |
| mobilenetv2_1.00_224 (Functional) | (None, 7, 7, 1280) | 2,257,984 |
| global_average_pooling2d (GlobalAveragePooling2D) | (None, 1280) | 0 |
| dense (Dense) | (None, 256) | 327,936 |
| dropout (Dropout) | (None, 256) | 0 |
| dense_1 (Dense) | (None, 5) | 1,285 |

Total params: 2,587,205 (9.87 MB)
Trainable params: 329,221 (1.26 MB)
Non-trainable params: 2,257,984 (8.61 MB)
Epoch 1/10

*Figuur 19 lagen transferlearning*

Na de eerste trainingsfase is fine-tuning toegepast. Bij fine-tuning worden niet alle lagen van het basisnetwerk vastgehouden, maar wordt een deel van de diepere lagen alsnog trainbaar gemaakt. Hierdoor kan het model zich verder aanpassen aan de specifieke Features van de Mondriaan-schilderijen.

Tijdens deze fase zijn extra lagen van MobileNetV2 vrijgegeven, waardoor het aantal trainable parameters toeneemt. De learning rate is hierbij lager gehouden om te voorkomen dat eerder geleerde kennis verloren gaat.

Model: "functional_1"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer (InputLayer) | (None, 224, 224, 3) | 0 |
| sequential (Sequential) | (None, 224, 224, 3) | 0 |
| true_divide (TrueDivide) | (None, 224, 224, 3) | 0 |
| subtract (Subtract) | (None, 224, 224, 3) | 0 |
| mobilenetv2_1.00_224 (Functional) | (None, 7, 7, 1280) | 2,257,984 |
| global_average_pooling2d (GlobalAveragePooling2D) | (None, 1280) | 0 |
| dense (Dense) | (None, 256) | 327,936 |
| dropout (Dropout) | (None, 256) | 0 |
| dense_1 (Dense) | (None, 5) | 1,285 |

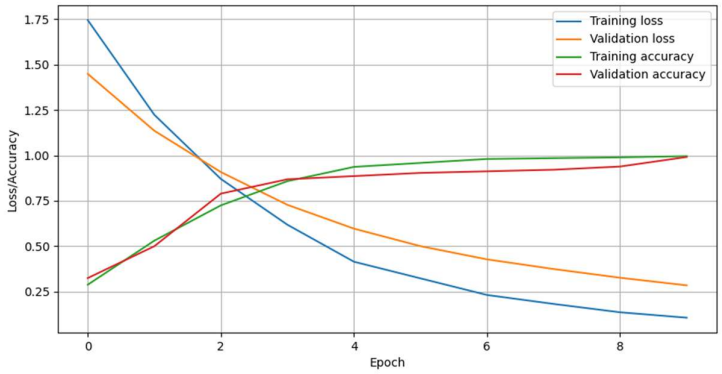
Total params: 2,587,205 (9.87 MB)
Trainable params: 1,954,117 (7.45 MB)
Non-trainable params: 633,088 (2.42 MB)
Epoch 1/5

*Figuur 20 meer lagen vrij gegeven*

Na toepassing van transfer learning en fine-tuning liet het model zeer goede prestaties zien op zowel de validatie- als testset. De validatienauwkeurigheid bereikte 100% en ook de testset liet een hoge nauwkeurigheid zien.

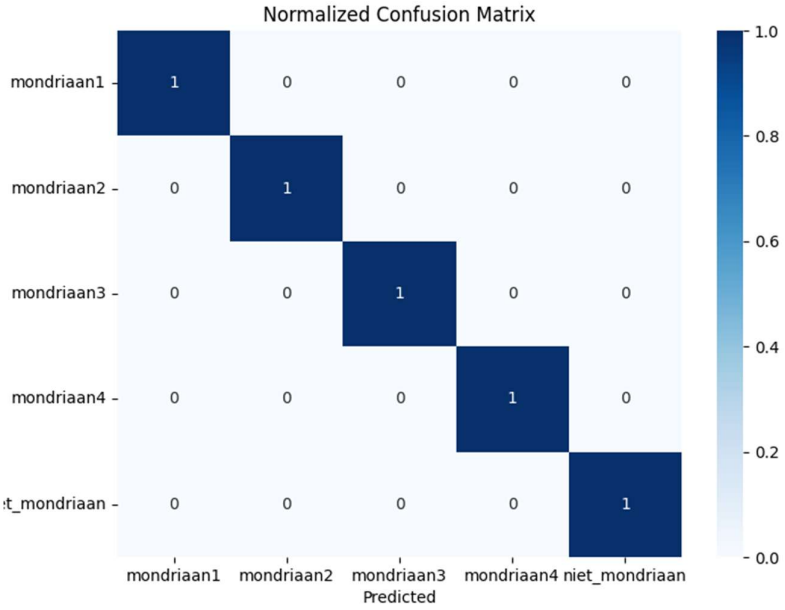
### 4.6.1 Resultaten

Figuur 21 toont de learning curves van het transfer learning model. Hierin is te zien dat zowel de trainings- als validatienauwkeurigheid snel toenemen en uiteindelijk dicht bij elkaar blijven. Tegelijkertijd nemen de training- en validatieloss gestaag af. Dit wijst erop dat het model goed leert zonder duidelijke tekenen van overfitting.




*Figuur 21 learning curve*

Figuur 22 toont de confusion matrix van het model op de validatie set. In deze matrix is te zien dat alle klassen correct worden geclassificeerd, zonder verwarring tussen de verschillende Mondriaan-klassen en de klasse niet_mondriaan. Dit resulteert in een nauwkeurigheid en F1-score van 1.00 op de validatie set.




*Figuur 22 confusion matrix*

De combinatie van de learning curve en de confusion matrix laat zien dat het transfer learning model zeer stabiel presteert en de verschillende klassen duidelijk van elkaar kan onderscheiden.

# 5 DEPLOY AND TEST

Voor de implementatie van het model is een softwaretoepassing opgezet die vele onderdelen van dit rapport gebruikt om een voorspelling te maken. De toepassing zal het volgende doen:

- Inladen van een directory met verschillende afbeeldingen of juist met een afbeelding.
- De afbeelding verkleinen tot de grootte waar het model mee om kan gaan (224,224).
- De afbeelding wordt omgezet naar een array.
- Voorspelling doen met behulp van het model.
- Zekerheid voorspellen met behulp van het model.
- Weergave van voorspelling aan de gebruiker.

Dit programma is in staat om een enkele afbeelding in te laden, maar ook een complete directory. De mogelijkheid om gebruik te maken van een webcam is nu niet geïmplementeerd, dit omdat er getest wordt met een telefooncamera en niet met een laptopcamera. Deze toepassing zou in de toekomst nog gerealiseerd kunnen worden.

De complete code voor het testen is terug te vinden in de meegeleverde code onder test_model.py.

## 5.1 Testplan

Dit project is ontwikkeld op basis van zowel functionele als niet-functionele specificaties. Om te verifiëren dat aan deze specificaties wordt voldaan, is een testplan opgesteld.

In dit hoofdstuk wordt het testplan beschreven. Hierbij worden alle specificaties afzonderlijk bekeken en geëvalueerd om te bepalen of er een test voor vereist is. Waar nodig worden bijbehorende testscenario's opgesteld om te controleren of aan de gestelde voorwaarden wordt voldaan.

### 5.1.1 Testplan functionele specificaties

| # | SPECIFICATIE | TESTPLAN |
|---|---|---|
| F1 | Het systeem moet de vier vooraf gedefinieerde schilderijen (Mondriaan 1, Mondriaan 2, Mondriaan 3 en Mondriaan 4) correct herkennen met een nauwkeurigheid van minimaal 60%. | F1 wordt getest door een testset te maken met afbeeldingen gemaakt door camera beeld en afbeeldingen die zelf geaugmenteerd zijn. Minimaal 80% van de afbeeldingen moeten goed voorspeld worden. |
| F2 | Na inladen van de afbelding moet het systeem binnen 4 seconden de titel van het schilderij zichtbaar tonen op het scherm van de gebruiker. | Er wordt een functie gemaakt die de tijd gaat bijhouden tijdens de voorspelling. |
| F3 | Wanneer een schilderij niet overeenkomt met een van de vier vooraf gedefinieerde Mondriaan-schilderijen, moet het systeem binnen 4 seconden het schilderij classificeren als zijnde niet Mondriaan. | Dit wordt op dezelfde manier getest als F2. |
| F4 | Het systeem functioneert betrouwbaar bij kantoorverlichting. | De testset van F1 worden foto's gebruikt met kantoor verlichting. Ook hiervoor moet 80% goed voorspeld worden. |
| F5 | Het systeem moet correcte herkenning uitvoeren bij kijkhoeken tot maximaal 20 | Voor een precisie test worden er 25 afbeeldingen door augmentatie onder een hoek van max 20% gebogen |

| | | |
|---|---|---|
| | graden ten opzichte van het schilderijoppervlak. | kunnen worden. Deze afbeeldingen moeten dan voor minimaal 80% correct voorspeld worden. |
| F6 | Het systeem moet betrouwbare herkenning uitvoeren bij een afstand tussen 30 cm en 50 cm van het schilderij. | Om dit te testen worden er 20 foto's gemaakt worden op afstanden tussen 30 en 50 cm. Van deze foto's moet 80% correct voorspeld worden. |

### 5.1.2 Testplan non functionele specificaties

| # | **SPECIFICATIE** | TESTPLAN |
|---|---|---|
| NF1 | Het systeem moet een afbeelding kunnen inladen van JPEG-formaat (.jpg). | Uitleg code |
| NF2 | Het systeem moet alle ingevoerde afbeeldingen automatisch schalen naar een resolutie van 1920×1080 pixels (Full HD) vóór verdere verwerking, om een consistente beeldkwaliteit en verwerkingssnelheid te garanderen. | Uitleg code. Alle afbeeldingen worden naar een kleiner formaat verkleint (224, 224). Dit met de rede dat de CNN hier beter mee om kan gaan. |
| NF3 | Het getrainde model moet bij validatie met een onafhankelijke testdataset een herkenningsnauwkeurigheid van minimaal 80% behalen voor de vier vooraf gedefinieerde schilderijen. | Dit wordt samen met F1 getest. |
| NF4 | De trainingsdataset moet bestaan uit minimaal 100 gelabelde afbeeldingen per schilderij, met variatie in hoek en afstand om realistische omstandigheden te simuleren. | Zie dataset, alleen uitleg vereist. |
| NF5 | Het systeem moet gebruik maken van een CNN. | Zie code, alleen uitleg vereist. |
| NF6 | Het systeem moet bij elke voorspelling een zekerheidsscore genereren tussen 0 en 100%, die de betrouwbaarheid van de classificatie aangeeft. | Dit is een could, alleen uitleg vereist. |
| NF7 | Wanneer de zekerheidsscore van een voorspelling lager is dan 50%, moet het systeem automatisch de gebruiker instrueren om één of meerdere extra beelden te maken voor een nieuwe analyse. | Dit is een could, alleen uitleg vereist. |

## 5.2 Uitvoeren testplan

In dit hoofdstuk worden de resultaten gepresenteerd van het testplan dat is opgesteld in hoofdstuk 5.1. De resultaten worden per onderwerp behandeld. De testen worden afzonderlijk voor de CNN uitgevoerd als transfer learning.

transferlearning = TFL

### 5.2.1 7.3.1 Precisie

Voor functionele specificatie 1 moet het systeem in staat zijn om de verschillende Mondriaan-schilderijen te herkennen met een nauwkeurigheid van minimaal 80%.

Daarnaast geldt voor niet-functionele specificatie 3 dat een minimale nauwkeurigheid van 80% behaald moet worden bij het gebruik van een onafhankelijke dataset.

| # | SPECIFICATIE | DL | UITSLAG |
|---|---|---|---|
| F1 | Het systeem moet de vier vooraf gedefinieerde schilderijen (Mondriaan 1, Mondriaan 2, Mondriaan 3 en Mondriaan 4) correct herkennen met een nauwkeurigheid van minimaal 60%. | CNN | Totaal aantal voorspellingen: 131<br>Goed voorspeld: 105<br>Nauwkeurigheid = 80.15% |
| F4 | Het systeem functioneert betrouwbaar bij kantoorverlichting. | TFL | Totaal aantal voorspellingen: 131<br>Goed voorspeld: 111<br>Nauwkeurigheid = 85% |
| NF3 | Het getrainde model moet bij validatie met een onafhankelijke testdataset een herkenningsnauwkeurigheid van minimaal 80% behalen voor de vier vooraf gedefinieerde schilderijen. | | |

Zoals uit de resultaten blijkt, doet de tranferlearning het over het algemeen net iets beter dan de CNN. Dit valt te verklaren omdat de transferlearning al langer en beter getraind is.

### 5.2.2 Timing

Voor functionele specificaties F2 en F3 moet de uitvoering van de software binnen een bepaald tijdsbereik plaatsvinden. Om dit te kunnen meten, is een hulpfunctie ontwikkeld die de uitvoeringstijd registreert.

| # | SPECIFICATIE | DL | UITSLAG |
|---|---|---|---|
| F2 | Na inladen van de afbelding moet het systeem binnen 4 seconden de titel van het schilderij zichtbaar tonen op het scherm van de gebruiker. | beidde | Resultaten liggen tussen de 100 en 300 ms bij Mondriaan en Niet Mondriaan |
| F3 | Wanneer een schilderij niet overeenkomt met een van de vier vooraf gedefinieerde Mondriaan-schilderijen, moet het systeem binnen 4 seconden het schilderij classificeren als zijnde niet Mondriaan. | | |

Aangezien de gemeten verwerkingstijd ruim binnen de gespecificeerde limiet valt, kan worden geconcludeerd dat dit doel is behaald.

### 5.2.3 Kijkhoek

Voor functionele specificatie 5 moet het systeem in staat zijn om afbeeldingen die onder een kijkhoek van 20 graden zijn genomen correct te classificeren. Voor deze test zijn vijf afbeeldingen per klasse geselecteerd. Deze afbeeldingen zijn vervolgens met behulp van data-augmentatie onder een hoek van 20 graden geroteerd.

| # | SPECIFICATIE | DL | UITSLAG |
|---|---|---|---|
| F5 | Het systeem moet correcte herkenning uitvoeren bij kijkhoeken tot maximaal 20 graden ten opzichte van het schilderijoppervlak. | CNN | Totaal aantal voorspellingen: 25<br>Goed voorspeld: 20<br>Nauwkeurigheid = 84% |
| | | TFL | Totaal aantal voorspellingen: 25<br>Goed voorspeld: 21<br>Nauwkeurigheid = 80% |

De nauwkeurigheid ligt op minimaal 80%. Dit doel is dus behaald. Hierin is te zien dat de CNN beter scoort dan de transfer learning. Dit heeft te maken met de augmentatie laag die goed zijn werk doet.

### 5.2.4 Kijkafstand

Voor de kijkafstand geldt dat de afbeeldingen correct moeten worden geclassificeerd bij een afstand van 30 tot 50 cm van het schilderij. Voor deze test zijn nieuwe foto's gemaakt met telefooncamera, maar met variërende afstanden tussen 30 en 50 cm. Om aan de specificatie te voldoen, moet minimaal 80% van de afbeeldingen correct worden geclassificeerd.

| # | SPECIFICATIE | DL | UITSLAG |
|---|---|---|---|
| F6 | Het systeem moet betrouwbare herkenning uitvoeren bij een afstand tussen 30 cm en 50 cm van het schilderij. | Beidde | Totaal aantal voorspellingen: 31<br>Goed voorspeld: 30<br>Nauwkeurigheid = 96.75% |

OP de kijkhoek scoort het model goed, met dit percentage is het doel behaald. Deze code werkt naar behoren.

### 5.2.5 Beste model

Aan de hand van de testscore blijkt de transferlearning net beter te scoren dan de CNN. Echter is de CNN helemaal zelf opgebouwd en doet niet veel onder aan de transferlearning. Beide modellen functioneren goed en zijn alle testen goed doorgekomen.

### 5.2.6 Uitleg code

Om aan de laatste specificaties te voldoen is alleen uitleg van de code benodigd. Voor een duidelijke uitleg bekijk de code. In de code is commentaar aanwezig die duidelijkheid verschaft.

# 6 CONCLUSION

In dit project is een deeplearning systeem ontwikkeld voor het herkennen van schilderijen van Piet Mondriaan. Het doel was om een werkend prototype te realiseren dat onder realistische omstandigheden, zoals variaties in licht, kijkhoek en afstand Mondriaan-schilderijen correct kan classificeren. Hierbij is zowel een zelfontworpen Convolutional Neural Network (CNN) als een transfer learning-aanpak met MobileNetV2 onderzocht en geëvalueerd.

Het project is uitgevoerd volgens een gestructureerde workflow. Eerst zijn de functionele en niet-functionele eisen opgesteld op basis van een SMART-probleemdefinitie. Vervolgens is een dataset samengesteld en uitgebreid met behulp van data-augmentatie om realistische museumomstandigheden te simuleren. Daarna is een preprocessing pipeline opgezet om de afbeeldingen geschikt te maken voor invoer in een CNN. Op basis hiervan zijn twee modellen getraind: een custom CNN en een transfer learning-model met MobileNetV2. Tot slot zijn beide modellen gedeployed en getest aan de hand van een uitgebreid testplan.

De belangrijkste ontwerpkeuzes waren het gebruik van relatief grote kernels (7×7) om de geometrische structuren van Mondriaan-schilderijen beter te herkennen, het toepassen van dropout om overfitting te voorkomen en het inzetten van data-augmentatie om de generaliseerbaarheid van het model te vergroten. Daarnaast is gekozen voor MobileNetV2 vanwege de goede balans tussen prestaties en rekenefficiëntie.

De behaalde resultaten laten zien dat de oorspronkelijke doelen grotendeels zijn gehaald. De functionele eis om de schilderijen met minimaal 60% nauwkeurigheid te herkennen is ruimschoots behaald: de CNN behaalde circa 80% en het transfer learning-model circa 85% nauwkeurigheid op realistische testdata. Ook de niet-functionele eis van minimaal 80% nauwkeurigheid op een onafhankelijke dataset is gehaald. De verwerkingstijd bleef met 100–300 ms ruim binnen de gestelde limiet van 4 seconden. Daarnaast functioneerde het systeem betrouwbaar bij kantoorverlichting, verschillende kijkhoeken tot 20 graden en kijkafstanden tussen 30 en 50 cm.

Wat betreft generalisatieprestaties laten zowel de learning curves als de testresultaten zien dat de modellen goed omgaan met nieuwe, ongeziene data. De validatie- en testprestaties liggen dicht bij elkaar en er zijn geen duidelijke tekenen van overfitting. Dit wordt ondersteund door de confusion matrices, waarin slechts beperkte verwarring tussen klassen optreedt. Het transfer learning-model presteert hierbij iets beter dan het custom CNN, wat verklaard kan worden door de vooraf geleerde algemene beeldfeatures uit ImageNet.

Terugkijkend op de aanpak kan worden geconcludeerd dat data-augmentatie, dropout en transfer learning zeer effectief zijn gebleken voor deze toepassing. Een aandachtspunt is dat de dataset relatief beperkt blijft en vooral bestaat uit gecontroleerde beelden. Voor toekomstige projecten zou het verzamelen van meer echte museumbeelden en het gebruik van een live camera feed de robuustheid verder kunnen verbeteren. Daarnaast kan onderzoek naar verdere optimalisatie voor embedded hardware een volgende stap zijn.

Samenvattend kan worden gesteld dat dit project succesvol aantoont dat deep learning geschikt is voor het herkennen van Mondriaan-schilderijen. De opgedane kennis en ervaring vormen een solide basis voor toepassing van embedded vision en machine learning binnen het main project van de minor.

# 7 LIJST AFBEELDINGEN

# 8 REFERENCES

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Sebastopol, Canada.: O'Reilly Media.

*SMART criteria*. (2020, 05 14). Retrieved from wikipedia: https://en.wikipedia.org/wiki/SMART_criteria

van Zwol, J., Hakimi, S., & van Eeten, R. (2025). *Research Rapport.* Arnhem: HAN.

# CODE APPENDICES

Door het gehele document zijn verschillende code appendices toegevoegd. De locaties hiervan zijn terug te vinden in de afbeeldingen lijst van Hoofdstuk 7.

# 9 BIJLAGE

## 9.1 Bijlage A

Search: Running Trial #30

| Value | Best Value So Far | Hyperparameter |
|---|---|---|
| 16 | 24 | filters_1 |
| 32 | 48 | filters_2 |
| 128 | 64 | filters_3 |
| 0.3 | 0.4 | dropout |
| 128 | 128 | units |
| 0.0001 | 0.001 | learning_rate |
| 10 | 10 | tuner/epochs |
| 0 | 4 | tuner/initial_epoch |
| 0 | 2 | tuner/bracket |
| 0 | 2 | tuner/round |

Trial 30 Complete [00h 03m 04s]

val_accuracy: 0.831932783126831

Best val_accuracy So Far: 0.9747899174690247

Total elapsed time: 00h 40m 44s

The hyperparameter search is complete. The optimal number of units in the first densely-connected layer is 128 and the optimal learning rate for the optimizer is 0.001.

Search space summary

Default search space size: 6

filters_1 (Int)

{'default': None, 'conditions': [], 'min_value': 8, 'max_value': 32, 'step': 8, 'sampling': 'linear'}

filters_2 (Int)

{'default': None, 'conditions': [], 'min_value': 16, 'max_value': 64, 'step': 16, 'sampling': 'linear'}

filters_3 (Int)

{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 128, 'step': 32, 'sampling': 'linear'}

dropout (Float)

{'default': 0.2, 'conditions': [], 'min_value': 0.2, 'max_value': 0.5, 'step': 0.1, 'sampling': 'linear'}

units (Int)

{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 128, 'step': 32, 'sampling': 'linear'}

learning_rate (Choice)

{'default': 0.01, 'conditions': [], 'values': [0.01, 0.001, 0.0001], 'ordered': True}

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| sequential (Sequential) | (None, 224, 224, 3) | 7 |
| rescaling (Rescaling) | (None, 224, 224, 3) | 0 |
| normalization (Normalization) | (None, 224, 224, 3) | 7 |
| conv2d (Conv2D) | (None, 218, 218, 24) | 3,552 |
| max_pooling2d (MaxPooling2D) | (None, 109, 109, 24) | 0 |
| conv2d_1 (Conv2D) | (None, 103, 103, 48) | 56,496 |
| max_pooling2d_1 (MaxPooling2D) | (None, 51, 51, 48) | 0 |
| conv2d_2 (Conv2D) | (None, 45, 45, 64) | 150,592 |
| max_pooling2d_2 (MaxPooling2D) | (None, 22, 22, 64) | 0 |
| dropout (Dropout) | (None, 22, 22, 64) | 0 |
| flatten (Flatten) | (None, 30976) | 0 |
| dense (Dense) | (None, 128) | 3,965,056 |
| dense_1 (Dense) | (None, 5) | 645 |

Total params: 4,176,355 (15.93 MB)

Trainable params: 4,176,341 (15.93 MB)

Non-trainable params: 14 (64.00 B)

--- Model Performance Report ---

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| mondriaan1 | 0.96 | 1.00 | 0.98 | 24 |
| mondriaan2 | 1.00 | 0.94 | 0.97 | 16 |
| mondriaan3 | 1.00 | 1.00 | 1.00 | 22 |

| | | | | |
|---|---|---|---|---|
| mondriaan4 | 0.94 | 1.00 | 0.97 | 32 |
| niet_mondriaan | 1.00 | 0.92 | 0.96 | 25 |
| accuracy | | | 0.97 | 119 |
| macro avg | 0.98 | 0.97 | 0.98 | 119 |
| weighted avg | 0.98 | 0.97 | 0.97 | 119 |

Gemiddelde F1-score: 0.9746