DFML Projektarbeit - Bildverarbeitung

Vorstellung der Ergebnisse

Bildverarbeitung – Klassifikation für Satellitenbilder

- SpaceEye Dataset
  - 1750 Satellitenaufnahmen Miami Küste
  - ca. 1400 x 1400 Auflösung
  - RGB + NIR Kanal
  - Metadata.csv
  - 50:50 ships/non_ships
  - Vorverarbeitung:
    - Reflectance corrected
    - Color corrected
- Aufgabe:
  - Training und Test eines binary image classifier
  - Vorgaben zu Evaluierung:
    - Accuracy
    - ROC curve
    - AUC
    - F1 score

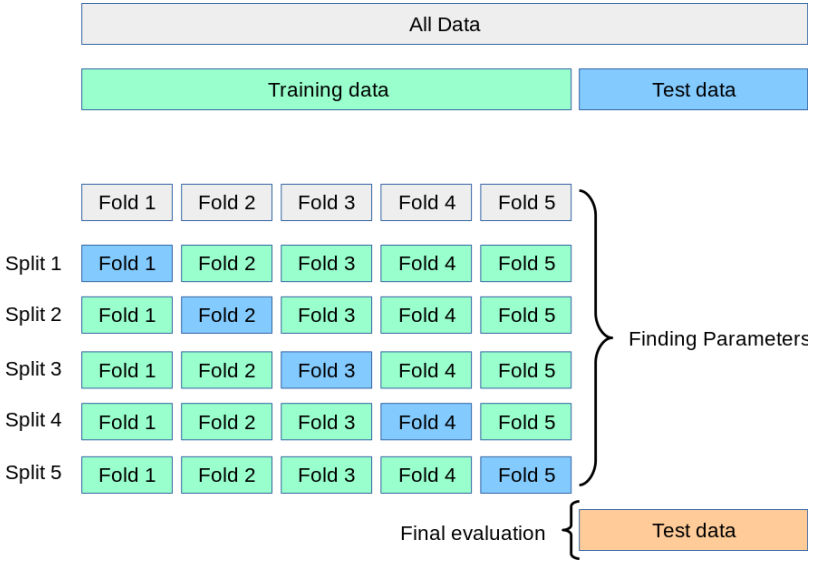
Data Preprocessing

- Anforderungen:
  - Dateiformat anpassen
  - Dataset Struktur erstellen
  - Split in Train und Test Daten
  - Cropping
  - **Resizing**
  - Normalization

- Lösungsschritte
  - .tif → .png/.jpeg (einlesbar für TF)
  - Aus <n>Ordner → <n>.png Dateien
  - 80 20 Train Test SplitCropping übernommen
  - Resizing 224 x 224 (VGG-16)
  - Normalization mit open-cv

Implementierter Algorithmus

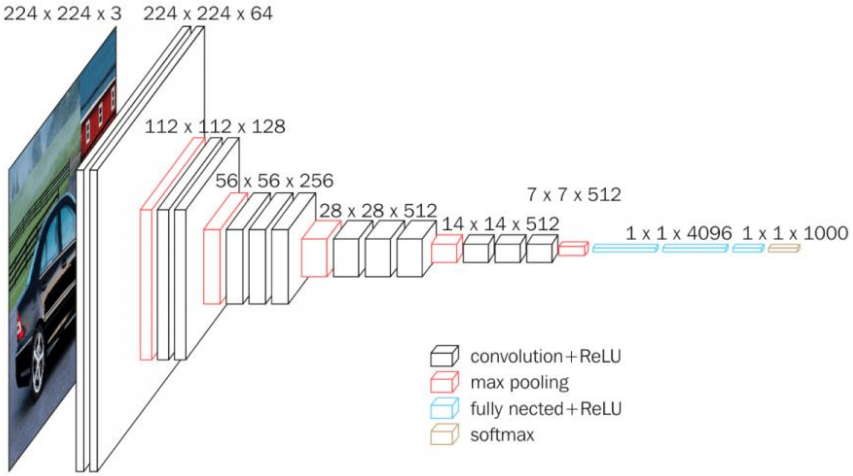
- 5-Fold-Cross-Validation
  - Splitten des Train Dataset in 5 Gruppen
  - Model Training mit 4
  - Model Validation mit 1
  - Wechsel der Gruppen in nächste Iteration
  - Stratified:
    - Label gleichverteilt in jeder Gruppe



Quelle: https://scikit-learn.org/stable/modules/cross_validation.html

Implementierter Algorithmus (RGB only)

- VGG-16 als pretrained model
  - weights von Imagenet, non trainable
- Binary classifier als fully connected Teil
  - Flatten layer
  - 2 Dense layer
  - Dropout

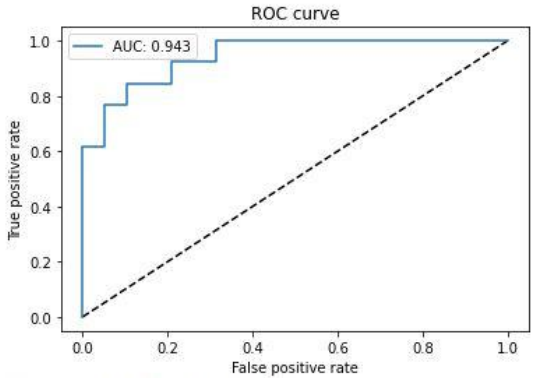


Quelle: https://neurohive.io/en/popular-networks/vgg16/

Implementierter Algorithmus (RGB only)

- Hyperparameter
  - Epochs = 10
  - Batchsize = 64
  - Optimizer = adam
    - Sehr einflussreich
    - Vgl. mit RMSProp

- Evaluation
  - Accuracy ~ 81%
  - F1 Score ~ 79%
  - AUC ~ 0.943




```
F1 Score: 0.7896961569786072
Validation accuracy: 0.8148148059844971
```

Implementierter Algorithmus (RGBN) - Probleme

- Data loading mit 4 Kanälen

→zuerst Versuch über .npy files

→kein Tensorflow Data Augmentation Unterstützung

→eigener Datasetgenerator notwendig

→.png format unterstützt rbga

→Nutzung des 4. Kanal für NIR

- Pretrained Models unterstützen meist nur RGB Bilder

→Input 4 Kanäle

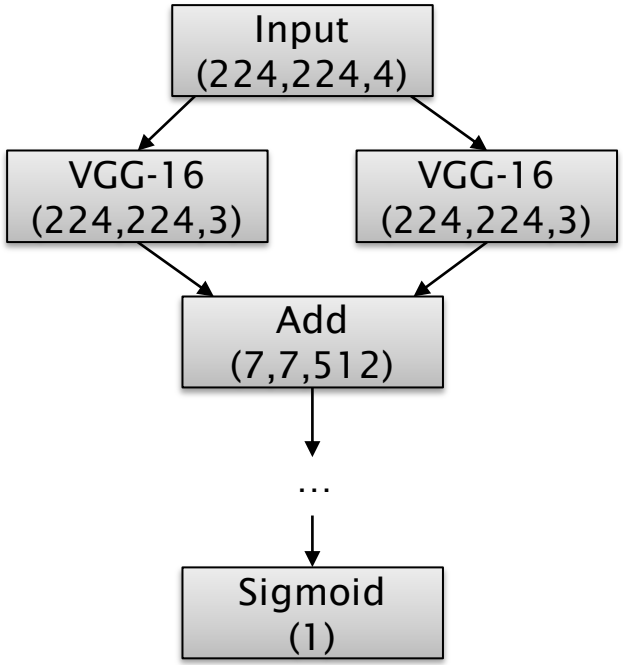
→split in RGB und NIR Teil

→jeweils eigenes VGG-16

→addiere output

→fully connected layer

Implementierter Algorithmus (RGBN)



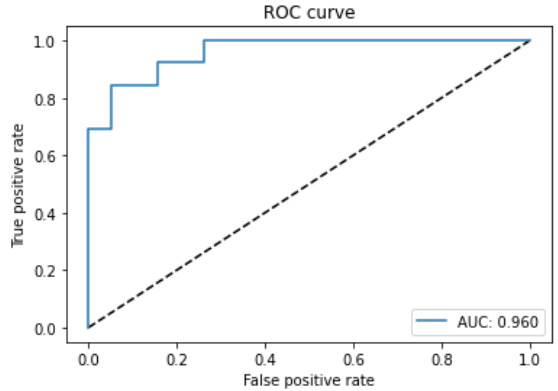
Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
input_10 (InputLayer)	[(None, 224, 224, 4)]	0	[]
lambda_6 (Lambda)	(None, 224, 224, 3)	0	['input_10[0][0]']
lambda_7 (Lambda)	(None, 224, 224)	0	['input_10[0][0]']
sequential_6 (Sequential)	(None, 7, 7, 512)	14714688	['lambda_6[0][0]']
sequential_7 (Sequential)	(None, 7, 7, 512)	14714688	['lambda_7[0][0]']
add_3 (Add)	(None, 7, 7, 512)	0	['sequential_6[0][0]', 'sequential_7[0][0]']
flatten_3 (Flatten)	(None, 25088)	0	['add_3[0][0]']
dense_8 (Dense)	(None, 512)	12845568	['flatten_3[0][0]']
dropout_5 (Dropout)	(None, 512)	0	['dense_8[0][0]']
dense_9 (Dense)	(None, 512)	262656	['dropout_5[0][0]']
dropout_6 (Dropout)	(None, 512)	0	['dense_9[0][0]']
dense_10 (Dense)	(None, 1)	513	['dropout_6[0][0]']

Total params: 42,538,113
Trainable params: 13,108,737
Non-trainable params: 29,429,376

Implementierter Algorithmus (RGBN)

- Evaluation
  - Accuracy ~ 86%
  - F1Score ~ 0.84
  - AUC ~ 0.96
- Interpretation
  - NIR Kanal verbessert Ergebnis sehr
  - K-Fold-Cross-Validation verhindert Overfitting zuverlässig
  - AUC sehr hoch →SpaceEye
  - Teil der Ungenauigkeit auch in Daten → dennoch mehr Training sinnvoll




```
F1 Score: 0.8392488360404968
Validation accuracy: 0.8644067645072937
```

Fazit und Ausblick

- Anfangsschwierigkeiten
- Sehr viel Trainingszeit
- Dennoch viel Spaß
- Projektziel sehr spannend

- Solider Ansatz für Training
- Definitiv noch Potential
  - Datenvorverarbeitung
  - Eigenes Model
  - Bessere Hyperparameteranpassung
  - Verwendung der metadaten