UNIVERSIDADE DE ÉVORA
ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Assignment 2 – "Image Classification"

Redes Neuronais e Aprendizagem Profunda

Miguel Silvério m55661 | Mestrado em Engenharia Informática
Ricardo Dias 59196 | Mestrado em Inteligência Artificial e Ciência de Dados
Évora 2023/2024

Contents

Introduction

The goal of this project was to create a model capable of estimating the Aerosol Optical Thickness (AOT) at 550 nm for specific locations using Sentinel-2 satellite images.

To accomplish this goal, we utilized machine learning techniques to deploy a model capable of analyzing the given images and then predict the closest values.

This involved implementing a methodology to preprocess the data and train a model to accurately predict AOT values from the satellite images.

Given the complexity and high-dimensional nature of the image data, we analyzed the available modeling architectures and ultimately decided on using a Convolutional Neural Network (CNN).

Dataset Overview

The dataset consists of 5 files related to predicting Aerosol Optical Thickness (AOT) at 550 nm using Sentinel-2 imagery:

Train Data

- File: Train
- Instances: 10,439
- Attributes: 10 (9 numeric attributes, 1 file)
- Description: Training dataset for model development.

Test Data

- File: Test
- Instances: 2,713
- Attributes: 10 (9 numeric attributes, 1 file)
- Description: Testing dataset for evaluating model performance.

Sample Submission (Random)

- File: Sample sub random
- Instances: 2,713
- Attributes: 2 (2 numeric attributes)
- Description: Sample submission file format with random values.

Sample Submission (Median)

- File: Sample sub median
- Instances: 2,713
- Attributes: 2 (2 numeric attributes)
- Description: Sample submission file format with median values.

Sample Submission (Mean)

- File: Sample sub mean
- Instances: 2,713
- Attributes: 2 (2 numeric attributes)
- Description: Sample submission file format with median values.

Experimental Set up

The first step in our experimental setup involves getting our data ready for analysis. We start by setting up the paths and loading the necessary CSV files to keep everything organized and easily accessible.

For loading the images, we use a library called rasterio to read the .tif files.

After loading the images, we normalize the data pixel value to 0 and 1 values. This step is crucial because it standardizes the inputs, making the training process smoother and more efficient.

Model Architecture

In the next phase of our experimental setup, we focus on designing the model architecture.

For the convolutional layers, we use Conv2D with ReLU activation to extract spatial features from the Sentinel-2 images. We sequentially apply layers with 64, 128, and 256 filters to capture a variety of patterns in the data. Batch normalization is included to stabilize training and aid in convergence. To manage the spatial dimensions of the feature maps, we use MaxPooling2D layers, which help retain important spatial information while reducing the computational load. Dropout layers are also implemented to prevent overfitting by randomly disabling a fraction of input units during training, which promotes better generalization.

Next, we configured these layers with 512, 256, and 128 units to capture complex relationships within the data. For the output layer, we use a single neuron with linear activation to predict the continuous AOT value.

To optimize the model, we employ the Adam optimizer, known for its adaptive learning rate mechanism, which is highly effective for training deep neural networks efficiently. Our chosen loss function is Mean Absolute Error (MAE), which directly measures the average magnitude of errors in the prediction of AOT values, providing a clear metric for model performance.

Convolutional Layers:

We use Conv2D with ReLU activation to extract spatial features from Sentinel-2 images. The layers employ 64, 128, and 256 filters sequentially to capture different patterns.

Batch Normalization:

Included to stabilize training, aiding in convergence.

Pooling Layers:

MaxPooling2D reduces the spatial dimensions of feature maps, helping to maintain important spatial information while reducing computational load.

Dropout:

Implemented to prevent overfitting by randomly disabling a fraction of input units during training, promoting generalization.

Dense Layers:

- These fully connected layers interpret the extracted features. We've chosen configurations with 512, 256, and 128 units to capture complex relationships in the data.

Output Layer:

- A single neuron with linear activation is used to predict the continuous AOT value.

Optimizer:

- We employ the Adam optimizer due to its adaptive learning rate mechanism, which is beneficial for training deep neural networks efficiently.
- Loss Function: Mean Absolute Error (MAE) serves as our loss function, directly measuring the average magnitude of errors in the prediction of AOT values.

Model Training

In this phase of our experimental setup, we focus on training the model.

We start with a train-validation split. Using an 80-20 split, we ensure that a substantial portion of the data is dedicated to training while keeping enough data aside to validate the model's performance on unseen samples. To maintain consistency and reproducibility, we set a random state for this split.

When training the model, we experimented with two different configurations: training for 30 epochs with a batch size of 300, and training for 100 epochs with a batch size of 100. The use of validation data during training is crucial as it helps us monitor the model's performance on unseen data, providing early insights into potential overfitting issues.

Model Evaluation

After training, we evaluated the model.

We start by plotting the training history displaying the MAE for the validation subset vs the MAE of the training subset over epochs.

By visualizing both training and validation MAE, we can see how the model learns over time, identify trends, and spot any potential overfitting or underfitting issues.

Predictions and Submission

Finally, we evaluate the model on the test subset to get a final check on its performance.

This step ensures that the model generalizes well to new and unseen data and makes correct predictions.

These predictions are then formatted according to the submission requirements and submitted for evaluation.

Decision-Making Process for Model Selection and Parameters Specification

We chose a Convolutional Neural Network (CNN) for this task because satellite images are high-dimensional data with a spatial structure, which CNNs are specifically designed to handle. CNNs can automatically learn spatial hierarchies of features, essential for capturing complex patterns in multi-band satellite data wich is the case with the given dataset.

Parameters Specification

Model Architecture:

Convolutional layers with 64, 128, and 256 filters allow the model to capture increasingly complex patterns efficiently.

A kernel size of 3x3 is chosen for its balance in capturing fine details and relevant patterns.

Using 'same' padding preserves image size.

Regularization Techniques:

Batch normalization stabilizes and accelerates training by normalizing the outputs of previous layers.

Dropout layers (with a 0.3 dropout rate) prevent overfitting by randomly dropping a fraction of the neurons during training.

Activation Function:

ReLU is used for its ability to mitigate the vanishing gradient problem where the gradients become too small and stop the wights from being updates.

Optimizer:

The Adam optimizer was selected for its adaptive learning rate capabilities, making it effective for training deep networks with sparse gradients and noisy data.

Loss Function:

Mean Absolute Error (MAE) was chosen for this regression task as it directly measures the average size of errors in predictions.

Training Settings:

300 and 100 where the chosen test epochs to ensure the model has enough time to learn from the data without overfitting.

A batch size of 30 and 100 were put to test to balance memory usage, convergence speed and obtain more stable MAE predictions due to the training history plot.

These decisions were made to ensure the model effectively captures the complexity of the satellite image data while maintaining efficient training and preventing overfitting.

MODELS

Parameters and Hyperparameters of the Models

Parameters

Convolutional Layers:

Conv2D Layer 1: 64 filters, kernel size of 3x3, ReLU activation, 'same' padding

Conv2D Layer 2: 128 filters, kernel size of 3x3, ReLU activation, 'same' padding

Conv2D Layer 3: 256 filters, kernel size of 3x3, ReLU activation, 'same' padding

Batch Normalization Layers:

Applied after each Conv2D layer to normalize and accelerate training.

Max Pooling Layers:

MaxPooling2D Layer 1: Pool size of 2x2

MaxPooling2D Layer 2: Pool size of 2x2

MaxPooling2D Layer 3: Pool size of 2x2

Dropout Layers:

Dropout Layer 1: Dropout rate of 0.3

Dropout Layer 2: Dropout rate of 0.3

Dropout Layer 3: Dropout rate of 0.3

Dropout Layer 4: Dropout rate of 0.3

Dropout Layer 5: Dropout rate of 0.3

Fully Connected (Dense) Layers:

Dense Layer 1: 512 units, ReLU activation

Dense Layer 2: 256 units, ReLU activation

Dense Layer 3: 128 units, ReLU activation

Output Layer:

Dense Output Layer: 1 unit (for regression) with linear activation

Hyperparameters

Optimizer:

Adam optimizer with a learning rate of 0.001.

Loss Function:

Mean Absolute Error (MAE).

Metrics:

Mean Absolute Error (MAE).

Training Configuration of Model 1 (submission 1):

Epochs: 300

Batch Size: 30

Training Configuration of Model 2 (submission 2):

Epochs: 100

Batch Size: 100

Validation Split:

Train-validation split ratio: 80-20

Random state for reproducibility: 20

Summary of the Models

Layer (type)	**Output Shape**	**Param #**
conv2d_1	(None, 19, 19, 64)	7,552
batch_normalization_1	(None, 19, 19, 64)	256
max_pooling2d_1	(None, 9, 9, 64)	0
dropout_1	(None, 9, 9, 64)	0
conv2d_2	(None, 9, 9, 128)	73,856
batch_normalization_2	(None, 9, 9, 128)	512
max_pooling2d_2	(None, 4, 4, 128)	0
dropout_2	(None, 4, 4, 128)	0
conv2d_3	(None, 4, 4, 256)	295,168
batch_normalization_3	(None, 4, 4, 256)	1,024
max_pooling2d_3	(None, 2, 2, 256)	0
dropout_3	(None, 2, 2, 256)	0
flatten	(None, 1024)	0
dense_1	(None, 512)	524,800
dropout_4	(None, 512)	0
dense_2	(None, 256)	131,328
dropout_5	(None, 256)	0
dense_3	(None, 128)	32,896
dropout_6	(None, 128)	0
dense_4	(None, 1)	129
Total params		**1,067,521**
Trainable params		**1,066,625**
Non-trainable params		**896**

Table 1- Neural Network Design.

Discussion on the Models Outputs

The objective was to build and submit models capable of estimating the Aerosol Optical Thickness (AOT) at 550 nm for a specific location using Sentinel-2 images. Here, we compare and discuss the results of two different model configurations:

Model 1 - Submission 1

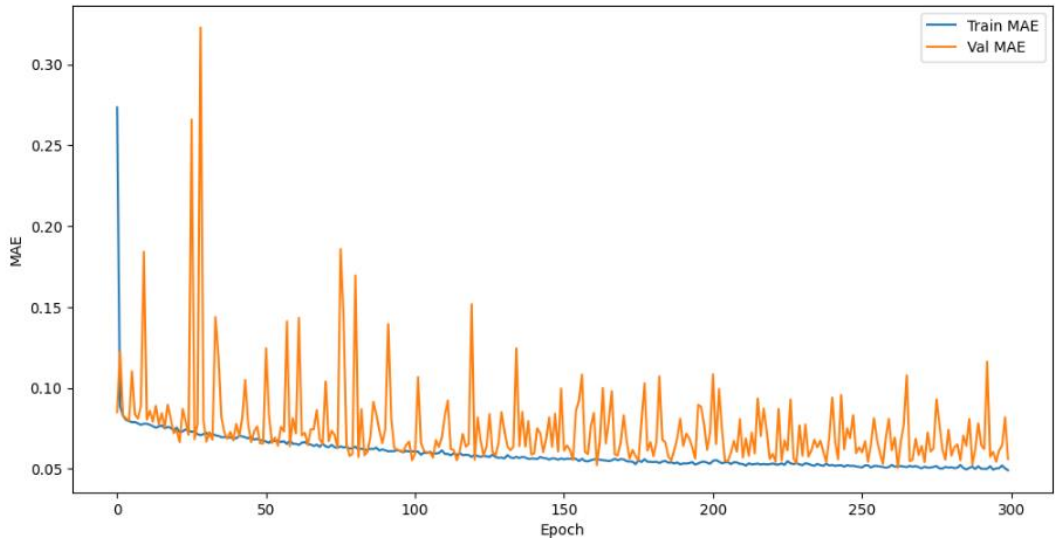

Figure 1- Training plot history of the Training subset MAE vs Validation subset MAE over 300 epochs with a batch size of 30.

Validation MAE obtained: 0.0560

Observations:

Training Curve:

The first training curve indicates a smooth decrease in Mean Absolute Error (MAE) over 300 epochs. This suggests that the model effectively learns and improves its predictions over a more extended training period.

Validation Curve:

The validation MAE exhibits fluctuations but shows a general trend of decreasing. This variability could indicate some overfitting, as the model might be capturing noise in the training data over such a long training duration.

Model 2 - Submission 2

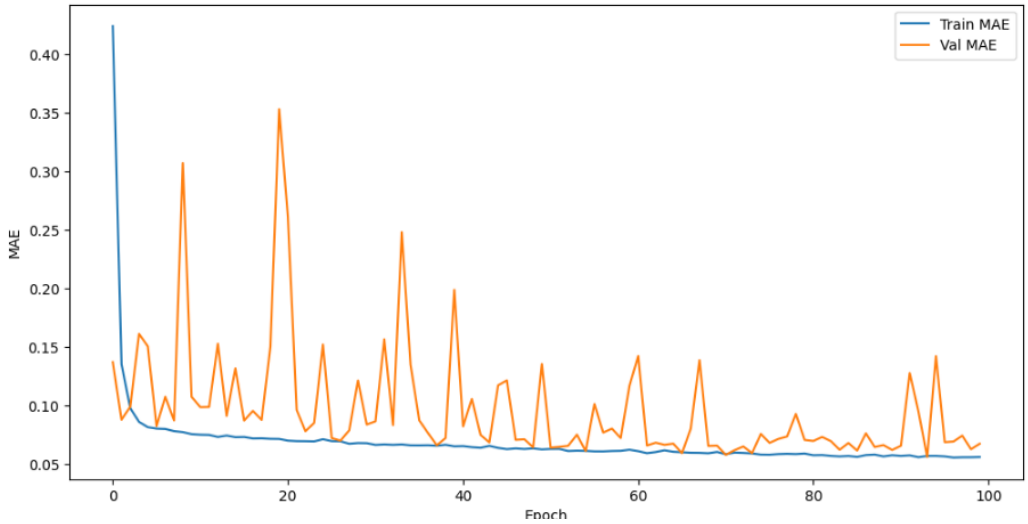

Figure 2 - Training plot history of the Training subset MAE vs Validation subset MAE over 100 epochs with a batch size of 100.

Validation MAE obtained: 0.0676

Observations:

Training Curve:

The training MAE decreases steadily, suggesting that the model is learning effectively.

Validation Curve:

The validation MAE shows more significant fluctuations than the first submission. The larger batch size of 100 might be causing the model to converge faster, but it might also be missing out on finer details that smaller batches could capture.

Results Discussion

When comparing the two different submissions, we noticed some differences in how they performed and were trained. The first submission achieved a MAE of 0.0560, which means it had better accuracy on new, unseen data compared to the second submission, which had a validation MAE of 0.0676. This suggests that the first model was generally more effective in making predictions on the test dataset data.

The first model underwent a longer training process of 300 epochs, and it used smaller batches of 30 at a time. This slower, more detailed approach likely allowed the

model to capture more subtle patterns in the data, leading to its better performance in validation. In contrast, the second model was trained for only 100 epochs and used larger batches of 100. While this sped up the training process, it may have caused the model to miss some of the details in the dataset. This could explain why its validation MAE was slightly higher.

Another observation was how unstable each model's validation MAE values were during training. The first model showed some higher fluctuations in its validation MAE over time, which could suggest it was occasionally overfitting the data. However, despite these fluctuations, it consistently performed better overall. The second model, even with a higher batch size also a lot of fluctuations on the MAE validation, indicating it might not have had enough time to fully stabilize its learning process.

Conclusion

In conclusion, the choice of epochs and batch size significantly impacts the model's performance. While the first model trained for more epochs and with a smaller batch size yielded better results, it also showed signs of overfitting. The second model, with fewer epochs and a larger batch size, was trained faster but had a slightly higher validation MAE.

Unfortunately since we didn’t started this project with the needed time and even though we only made two submissions we surpassed a lot of obstacles during this project, since the configuration of python with “tensorflow” that required an older python version and “miniconda” to utilize the GPU to make the calculations, to the data preprocessing phase that during the majority of the time was giving us test MAE values in the best of cases of 3 and we didn’t found that the problem was the way we were extraction the AOT values until a couple hours to the submission timeline. This last setback was the cause of us only making two submissions and we didn’t submit the other architectures tested for our design and even other model parameters and experimental iterations that could be relevant to discuss, learn and obtain a deeper understanding of the whole process of building a model.

Even with this partially successful experimentation and exploration of a model building process the whole project served as a good teaching tool for us as our first model building process from start to finish and helped to acquire a better understanding of the impact of different training configurations and guides adjustments to improve future models.

References

Yamashita, R., Nishio, M., Do, R.K.G. *et al.* Convolutional neural networks: an overview and application in radiology. *Insights Imaging* **9**, 611–629 (2018).