Djedjiga SALMI

Process:

To create a binary classification of “roads” and “fields” data we ca use a convolutional neural network (CNN) by following these steps:

- Preprocess the images by resizing it to a standard size and normalizing it by dividing it by 255, this will help the model to learn easily.
- Divide the data into training and validation sets.
- Transform this data to numpy array format to be processed by the CNN or create generators to be passed on the fil model.
- We can augment the training data by performing transformations, such as rotations, zooms and inversion when we have a small dataset. This helps to prevent overfitting.
- Train the model on the training set and evaluate its performance on the validation set
- Use a CNN model with convolutional layers and max-pooling to reduce the image dimension, a dropout layer to prevent overfitting (in some case we prefer not use this type of layers, so if it doesn’t give a good performances. Then, a fully connected layers which take the extracted features and transform it into an output that corresponds to the different classes.
- Use an output layer with an activation function “sigmoid” to predict the probability that the image belongs to each class. The relu activation also which is commonly used in neural networks because it is computationally efficient and easy to implement.
- Use the binary cross-entropy function which is used in binary classification problem.
- Use the trained model to predict new images (test_images and other externes images).

For the hyperparameters, we can use:

- The update rate of the model’s weights at each stage of training (learning rate approximately 0.01, 0.001 or 0.0001).
- The number of images that are used simultaneously in the network at each stage of the training (batch size).
- The number of epochs which is the number times the model traverses the dataset.
- The relu activation function which is very used in CNN.
- The optimizer like Adam used in binary classification.

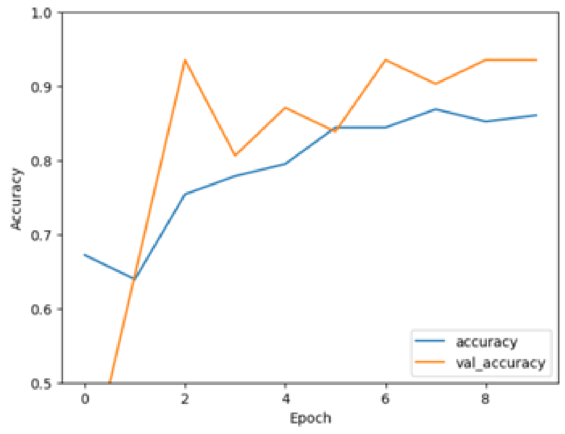
To evaluate the model, we can use metrics such as accuracy, the error, and test the model with other and new images which are not in the dataset.

Results:

After several attempts of changing the learning rate, and the complexity of the model by adding and deleting some layers, changing the learning rate value and the batch size. I get a good performance for 10 epochs and a learning rate of 0.0001.

Accuracy graph

```
1/1 - 1s - loss: 0.2725 - accuracy: 0.9355 - 642ms/epoch - 642ms/step
```



Loss graph

```
1/1 - 1s - loss: 0.2725 - accuracy: 0.9355 - 636ms/epoch - 636ms/step
```

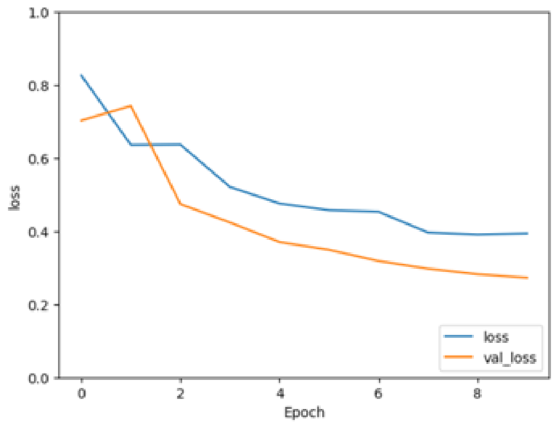


Figure 1 : Accuracy and loss graph

This graph shows that the performances of the model are good. It achieved a validation accuracy of 93.55% at the last epochs. It has been learned to to generalize the training and validation data well and can be used to predict the class of new images.

Test on the test_images:

Remember that the road class is presented by 1 and the fields class is presented by 0

Test on 1.jpeg

First, we have to resize and standardize this image and then predict its class using the formed model.

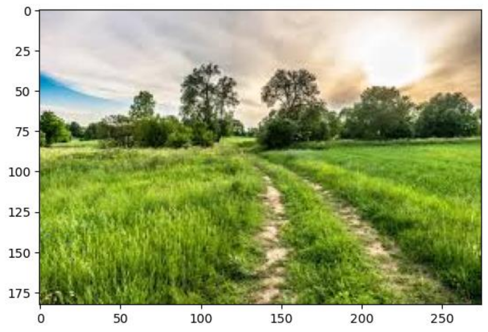


Figure 2:1.jpeg

Prediction result :

```
1/1 [==============================] - 0s 147ms/step
array([[0.08019949]], dtype=float32)
```

The value 0.08 is the probability of predicting the image. The model has correctly predicted the class of this image which is “fields”.

Test on 2.jpeg.

First, we have to resize and standardize this image and then predict its class using the formed model.

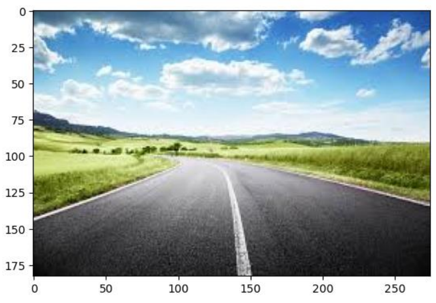


Figure 3 : 2.jpeg

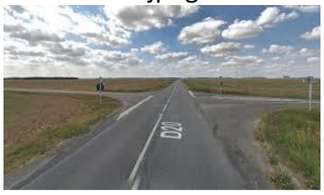
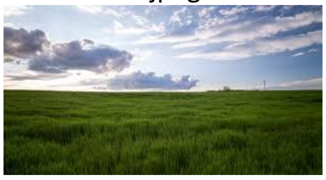
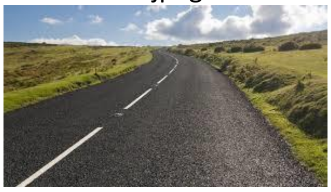
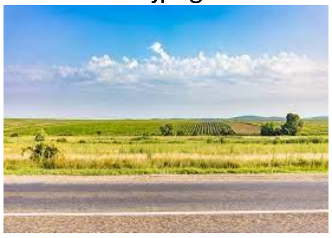
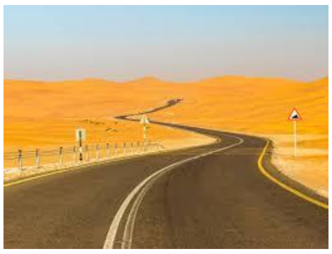
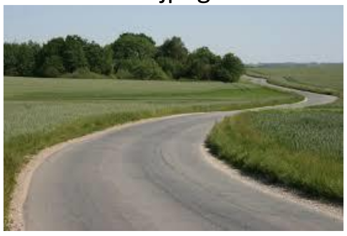
Prediction result:

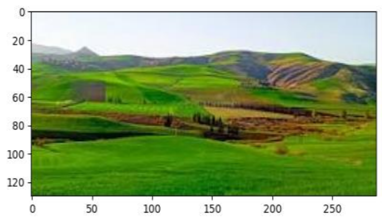
```
1/1 [==============================] - 0s 32ms/step
array([[0.9398777]], dtype=float32)
```

The value 0.9 is the probability of predicting the image. The model has correctly predicted the class of this image “roads”.

The table below summarize the results:

Image	Class prediction (0 to fields- 1 to roads)
1.jpeg	0.08
2.jpeg	0.94

3.jpeg 	0.90
4.jpeg 	0.42
5.jpeg 	0.72
6.jpeg 	0.64
7.jpeg 	0.36 (False)
8.jpeg 	0.79

9.jpeg	0.10
10.jpeg	0.39
extern	0.72
extern2 	0.08

Conclusion:

We have formed a model with a good performance. We have evaluated it with accuracy metric that g ave us a value of 222 and the loss metric of 222. The model has also recognized the images that are n ot in the dataset.