Scuola di Ingegneria
MsC in Artificial Intelligence and Deep Learning
Course of Computational Intelligence and Deep Learning

Self Autonomous Driving Car
Scene Parsing

Adelmo Brunelli

Anno Accademico 2022/2023

Contents

1 Introduction to Autonomous Driving and Scene Parsing

Autonomous driving represents a transformative leap in the field of transportation, aiming to revolutionize the way we navigate our roads and highways. This technological advancement holds the promise of safer, more efficient, and convenient transportation, where vehicles are capable of making critical driving decisions without human intervention. One of the key components enabling autonomous driving is the ability to perceive and understand the surrounding environment accurately. Scene parsing, a critical subfield of computer vision, plays a pivotal role in achieving this perception by analyzing the visual input from various sensors, such as cameras, lidar, and radar.

Scene parsing involves the semantic segmentation of the visual scene, which means assigning a specific label or category to each pixel in an image. This detailed understanding of the environment is essential for autonomous vehicles to navigate safely and make informed decisions. A crucial aspect of scene parsing is the detection and recognition of objects within the scene, which is often accomplished through the use of Convolutional Neural Networks (CNNs).

Convolutional Neural Networks are a class of deep learning models that have proven to be highly effective in a wide range of computer vision tasks, including object detection and recognition. These neural networks are designed to mimic the human visual system by applying a series of convolutional layers to extract meaningful features from the input images. In the context of autonomous driving and scene parsing, CNNs are trained on extensive datasets containing labeled images of various objects, road elements, and other relevant environmental features.

The ability of CNNs to detect and classify objects in real-world driving scenarios is a critical component of autonomous driving systems. These networks enable vehicles to recognize not only common objects like other vehicles, pedestrians, and traffic signs but also more complex and dynamic elements like road boundaries, lane markings, and obstacles.

Sensor fusion is a pivotal concept in autonomous driving systems. It enhances the capabilities of CNNs and other sensors by integrating data from multiple sources. This approach provides redundancy, improves perception, and enhances safety.

By integrating information from sensors such as cameras, LiDAR, radar, ultrasonic sensors, and GPS, autonomous vehicles gain a comprehensive view of their surroundings. Each sensor has its strengths and weaknesses, and sensor fusion ensures that the strengths of one sensor compensate for the weaknesses of another. For example, cameras provide high-resolution images, while LiDAR offers precise 3D point cloud data. Radar excels in adverse weather conditions, and ultrasonic sensors are ideal for short-range obstacle detection.

2 ApolloScape Scene Parsing (Semantic Segmentation) Dataset

The ApolloScape Scene Parsing (Semantic Segmentation) Dataset is a comprehensive collection of data resources designed to facilitate research and development in the field of semantic segmentation of scenes, a critical component for environmental perception in autonomous driving systems. This dataset is well-known for its extensive coverage and high-quality imagery.

2.1 Camera Front Scene Parsing Dataset

This folder contains thousands of images captured from the front camera of a vehicle in various driving conditions. The images were collected in a diverse range of scenarios, including urban environments, rural roads, highways, and various other driving situations. Each image comes with pixel-wise annotations, providing detailed information about semantic segmentation—assigning a specific category to each pixel in the image. These categories may include objects such as vehicles, pedestrians, road signs, buildings, and more.

2.2 Camera Stereo Scene Parsing Dataset

This folder contains stereo images captured by a pair of cameras mounted on the vehicle. This type of dataset is particularly useful for 3D vision applications. Similarly, these images are annotated for semantic segmentation, but the presence of two cameras enables better depth perception and a deeper understanding of objects in the scene.

In both folders, the images were captured under real-world conditions, including variations in lighting, diverse backgrounds, and complex driving scenarios. This reflects the real challenges that autonomous vehicles face on the road.

This dataset has been instrumental in training and evaluating algorithms for semantic segmentation and deep neural network models in the context of autonomous driving. By providing a detailed understanding of road scenes, it significantly contributes to the advancement of autonomous driving technologies and environmental perception.

Researchers and developers use this dataset to train and test their scene parsing algorithms, aiming to improve the accuracy and reliability of object detection and semantic segmentation in autonomous vehicles. It plays a pivotal role in advancing the state-of-the-art in Autonomous Driving systems and Computer Vision.

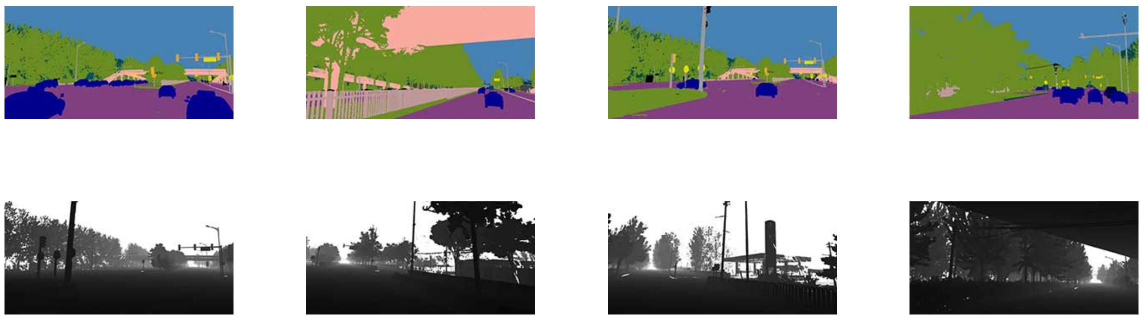


Figure 1: Eight Images in Two Rows

3 State of Art in Scene Parsing

Here are some notable works related to scene parsing in Computer Vision:

3.1 DeepLab

DeepLab is a deep learning model architecture that has made significant advancements in the field of semantic segmentation, a task that involves classifying each pixel in an image into a specific class or category. Developed by Google Research, DeepLab has achieved state-of-the-art results in various computer vision challenges, particularly in scene parsing and semantic segmentation tasks.

One of the key innovations introduced by DeepLab is the use of atrous convolutions, also known as dilated convolutions [5]. Atrous convolutions allow for the expansion of the model's receptive field (the area of the input image that a neuron in the network "sees") without downsampling the image and losing spatial resolution. This is crucial for semantic segmentation [6] tasks where fine-grained details in the image are important.

3.2 U-Net

U-Net is a powerful and widely-used deep neural network architecture designed for semantic segmentation tasks. It was introduced by Olaf Ronneberger, Philipp Fischer, and Thomas Brox in their 2015 paper titled "U-Net: Convolutional Networks for Biomedical Image Segmentation."[4] U-Net has become a cornerstone in various computer vision applications, especially in the medical field, but its versatility extends to other domains as well.

3.3 FCN (Fully Convolutional Network)

Fully Convolutional Networks (FCN) represent a pioneering approach in the field of computer vision and deep learning for solving pixel-wise semantic segmentation tasks. Introduced by Jonathan Long, Evan Shelhamer, and Trevor Darrell in their 2015 paper titled "Fully Convolutional Networks for Semantic Segmentation,"[3] FCN revolutionized the way we approach image segmentation.

3.4 Mask R-CNN

Mask R-CNN is a powerful and versatile deep learning architecture that extends the Faster R-CNN framework to incorporate pixel-wise segmentation in addition to object detection. It was introduced by Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick in the paper titled "Mask R-CNN" in 2017 [1]. This architecture has significantly contributed to tasks like object recognition, instance segmentation, and image analysis. [2]

3.5 Graph-based Approaches

Some works have explored the use of graph-based methods for semantic segmentation, enabling the capture of spatial relationships between pixels or points in the image.

These are just a few examples of related works in the field of scene parsing. Specific weighted-F1 scores would depend on the dataset, model configuration, and evaluation criteria used in each individual study.

4 Data Preprocessing for Semantic Segmentation Dataset

The provided code represents a data preprocessing phase for creating a dataset intended for semantic segmentation, specifically customized for the ApolloScape dataset. The primary goal is to prepare a dataset suitable for training machine learning models for semantic segmentation tasks, where each image is associated with a set of labels describing objects within the image.

4.1 Code Description

The primary steps in the code are as follows:

1. Define the directory paths for images and labels within the ApolloScape dataset. Specify a set of potential label values.
2. Initialize empty lists 'image' and 'rows' to store image file paths and associated labels.
3. Implement functions to collect image file paths and labels from JSON files.
4. Create a partial DataFrame by comparing each image with its associated labels, assigning binary values (1 or 0) to indicate the presence or absence of specific labels in each image.
5. Iterate through the directories of images and labels, gathering data and constructing the partial DataFrame.
6. Generate a complete DataFrame where each row corresponds to an image, and columns represent the presence of various object labels. Any missing label values are filled with 0.
7. Check for duplicate records based on the image ID.
8. Save the resulting DataFrame as a CSV file named "dataset.csv."

This code performs the initial data processing steps to prepare a structured dataset for semantic segmentation tasks. It is a crucial step in building machine learning models for tasks such as object detection and scene parsing in autonomous driving systems.

In our case, owing to the constraints of Google Colaboratory regarding GPU usage, fully leveraging the entire dataset was not feasible due to the extended training times associated with CPU-only computation. To overcome this challenge, we turned to scikit-learn's `train_test_split` function for dataset partitioning. Using this function, we randomly allocated images from the original training set into three distinct subsets: the training set, validation set, and test set. Our goal was to maintain the original dataset's data distribution while accommodating the limitations of our computing environment.

In our analysis of the extensive collection of photos totaling 21.64 gigabytes, we have encountered certain classes that exhibit a unique characteristic. Specifically, for a subset of these classes, the data samples in our dataset are represented as an array of all zeros. This occurrence arises due to the nature of the dataset and the inherent diversity in image content.

To address this issue and ensure the integrity of our analysis, we have made the decision to focus solely on the sixteen labels that are clearly indicated in the table provided at the end of our document. This approach allows us to mitigate the impact of empty or zero-valued samples, ensuring that our analysis is based on data that provides valuable insights and contributes to the overall objectives of our study.

5 CNN from scratch

In our quest for an effective solution, we delved into the realm of custom neural network architecture. Crafting the optimal architecture involves navigating a maze of hyperparameters, and we embraced a trial-and-error approach to discover the finest configuration. Our journey commenced with a modest model that struggled to find its footing. Gradually, we expanded its dimensions until it began to exhibit robust generalization.

Regarding the size of the input images, we decided to use 244×244 because of 6841 images. We chose to use 32 as batch size, which is the number of samples considered in each iteration. After the input layer, we performed a normalization with a Rescaling Layer, so that each pixel value was in the range [0,1]. Then we decided to use 5 convolutional layers, using the zero-padding technique, in order to give the same importance to each pixel of the image; in fact, even the borders are relevant having cropped the images as much as possible. For the stride we decided to use the default value of 1, to avoid penalties on the accuracy. As activation functions we used ReLU: $f(x) = \max(0, x)$. As the size of the local receptive fields, we decided to use the default value 3×3.

In the first convolutional layer 16 filters are used, and for each convolutional layer the number of filters doubles. Max-pooling is applied after each convolutional level, so that small regions are summarized with a single value. Our architecture uses increasing size for max pooling, with the size of the pooling increasing in the final levels (3x3 in the first layers and 5×5 in the last one). This allowed us to decrease the number of network parameters while minimizing accuracy loss. Indeed, we could also increase the stride or avoid zero-padding, but this solution is the one that achieves better performance.

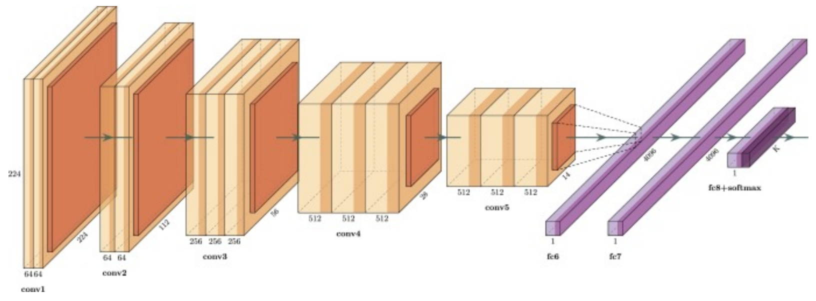


Figure 2: CNN architecture

The final levels change in the various experiments made. The only thing that does not change is the output layer, for which we used a dense layer with a 15 neurons, exploiting a sigmoid function to perform the multiclass classification. As a cornerstone of our regularization strategy, dropout emerged as the most adept technique, consistently delivering the best results for our unique problem.

In conclusion, our custom neural network architecture, coupled with thoughtful hyperparameter tuning, yielded promising results. The journey of crafting our model was marked by experimentation and refinement, ultimately leading to an architecture that demonstrates robust generalization.

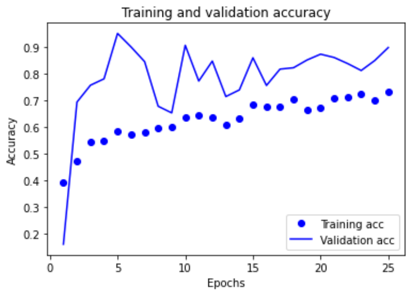
6 Experiments

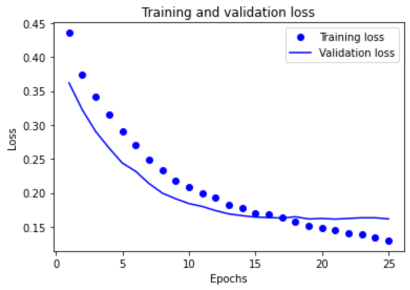
6.1 Experiment 1: one dense layer wth 256 neurons and dropout

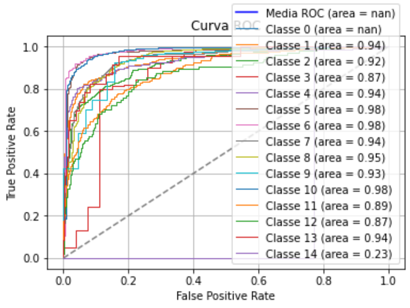
We try a simple model with a single dense layer of 256 neurons and dropout to fight the high overfitting.

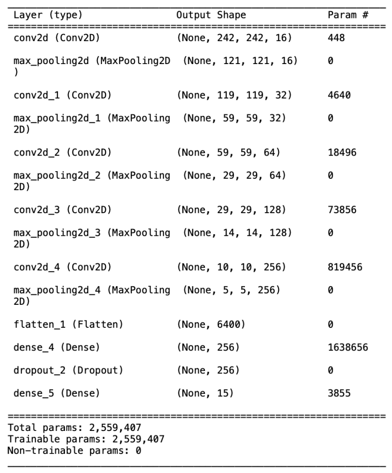
Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.1300	0.7333	0.1617	0.8990

Table 1: Training and validation statistics









```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 242, 242, 16)      448

 max_pooling2d (MaxPooling2D  (None, 121, 121, 16)     0
 )

 conv2d_1 (Conv2D)           (None, 119, 119, 32)      4640

 max_pooling2d_1 (MaxPooling  (None, 59, 59, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 59, 59, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 29, 29, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 29, 29, 128)       73856

 max_pooling2d_3 (MaxPooling  (None, 14, 14, 128)      0
 2D)

 conv2d_4 (Conv2D)           (None, 10, 10, 256)       819456

 max_pooling2d_4 (MaxPooling  (None, 5, 5, 256)        0
 2D)

 flatten_1 (Flatten)         (None, 6400)              0

 dense_4 (Dense)             (None, 256)               1638656

 dropout_2 (Dropout)         (None, 256)               0

 dense_5 (Dense)             (None, 15)                3855

=================================================================
Total params: 2,559,407
Trainable params: 2,559,407
Non-trainable params: 0
_________________________________________________________________
```

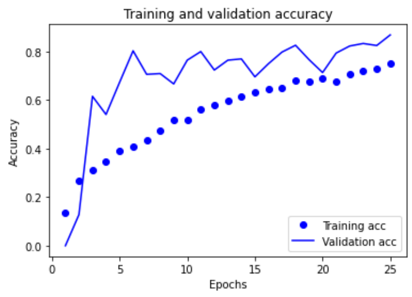
From the learning curves, it's evident that the model is not yet experiencing overfitting; it continues to generalize effectively on the validation set. While the validation loss is relatively low, there is room for improvement. The AUC metric is approximately 0.95, indicating high accuracy on the test dataset.

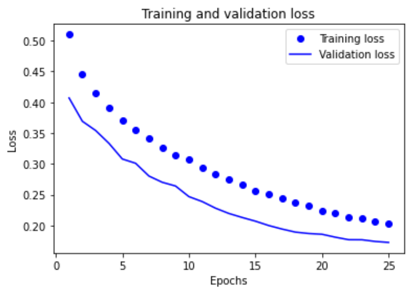
However, it's worth noting that around epoch 20, there seems to be a turning point where the risk of overfitting begins to emerge. This transition point is crucial to monitor as it may guide further optimization efforts to maintain model generalization while enhancing performance.

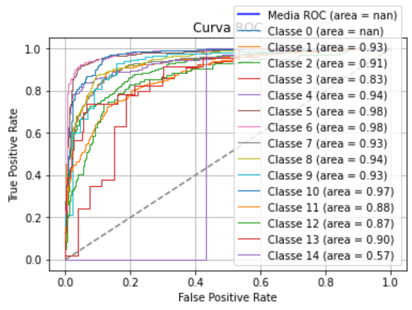
6.2 Experiment 2: two dense layer with 256 and 128 neurons and two dropout

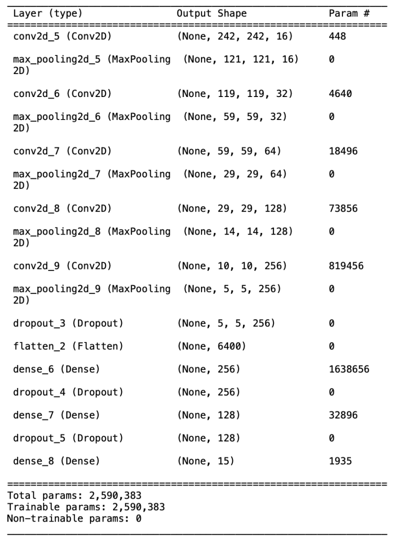
Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.2035	0.7500	0.1727	0.8687

Table 2: Training and validation statistics









```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_5 (Conv2D)           (None, 242, 242, 16)      448

 max_pooling2d_5 (MaxPooling  (None, 121, 121, 16)     0
 2D)

 conv2d_6 (Conv2D)           (None, 119, 119, 32)      4640

 max_pooling2d_6 (MaxPooling  (None, 59, 59, 32)       0
 2D)

 conv2d_7 (Conv2D)           (None, 59, 59, 64)        18496

 max_pooling2d_7 (MaxPooling  (None, 29, 29, 64)       0
 2D)

 conv2d_8 (Conv2D)           (None, 29, 29, 128)       73856

 max_pooling2d_8 (MaxPooling  (None, 14, 14, 128)      0
 2D)

 conv2d_9 (Conv2D)           (None, 10, 10, 256)       819456

 max_pooling2d_9 (MaxPooling  (None, 5, 5, 256)        0
 2D)

 dropout_3 (Dropout)         (None, 5, 5, 256)         0

 flatten_2 (Flatten)         (None, 6400)              0

 dense_6 (Dense)             (None, 256)               1638656

 dropout_4 (Dropout)         (None, 256)               0

 dense_7 (Dense)             (None, 128)               32896

 dropout_5 (Dropout)         (None, 128)               0

 dense_8 (Dense)             (None, 15)                1935

=================================================================
Total params: 2,590,383
Trainable params: 2,590,383
Non-trainable params: 0
_________________________________________________________________
```

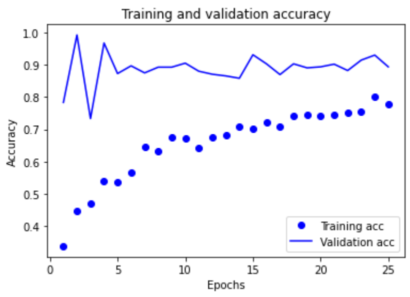
We observe that the network now achieves a slightly higher validation loss, specifically 0.1727, although it remains an excellent result. The AUC (Area Under the Curve) value of 0.92 indicates a high level of accuracy on the test dataset.

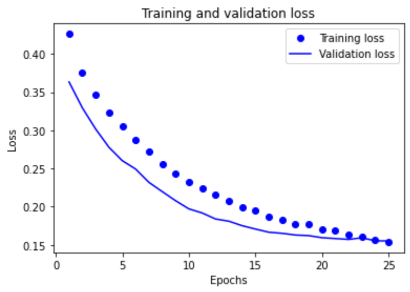
Interestingly, there isn't a distinct turning point in the curve where it appears to reverse (*turning point*), suggesting that there isn't significant evidence of overfitting. Consequently, the model exhibits an impressive validation loss, and the ROC curve shows good performance. This stability in performance without a noticeable turning point indicates the model's robustness and its ability to maintain strong generalization across various data points.

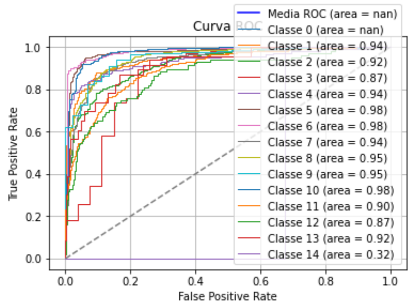
6.3 Experiment 3: one dense layer with 256 neurons and two dropouts

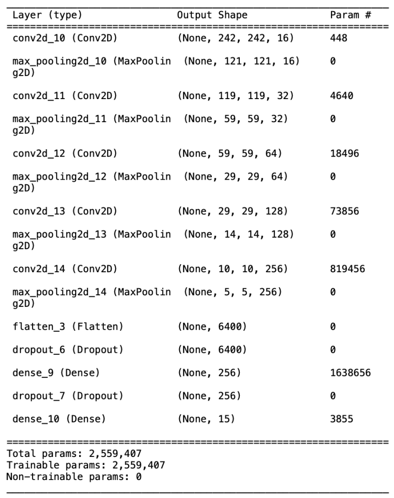
Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.1538	0.7792	0.1550	0.8938

Table 3: Training and validation statistics









```
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_10 (Conv2D)          (None, 242, 242, 16)      448       
                                                                 
 max_pooling2d_10 (MaxPoolin  (None, 121, 121, 16)     0         
 g2D)                                                            
                                                                 
 conv2d_11 (Conv2D)          (None, 119, 119, 32)      4640      
                                                                 
 max_pooling2d_11 (MaxPoolin  (None, 59, 59, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_12 (Conv2D)          (None, 59, 59, 64)        18496     
                                                                 
 max_pooling2d_12 (MaxPoolin  (None, 29, 29, 64)       0         
 g2D)                                                            
                                                                 
 conv2d_13 (Conv2D)          (None, 29, 29, 128)       73856     
                                                                 
 max_pooling2d_13 (MaxPoolin  (None, 14, 14, 128)      0         
 g2D)                                                            
                                                                 
 conv2d_14 (Conv2D)          (None, 10, 10, 256)       819456    
                                                                 
 max_pooling2d_14 (MaxPoolin  (None, 5, 5, 256)        0         
 g2D)                                                            
                                                                 
 flatten_3 (Flatten)         (None, 6400)              0         
                                                                 
 dropout_6 (Dropout)         (None, 6400)              0         
                                                                 
 dense_9 (Dense)             (None, 256)               1638656   
                                                                 
 dropout_7 (Dropout)         (None, 256)               0         
                                                                 
 dense_10 (Dense)            (None, 15)                3855      
                                                                 
=================================================================
Total params: 2,559,407
Trainable params: 2,559,407
Non-trainable params: 0
_________________________________________________________________
```

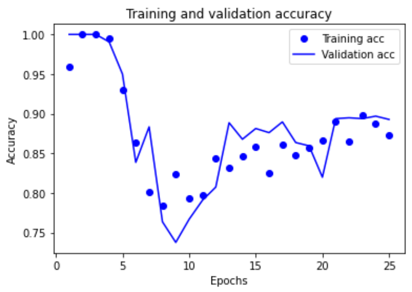
Here we can observe that we have achieved a lower validation loss value compared to the two previous cases. This represents the best model obtained thus far in the entire section on CNN from scratch. We can see that the validation loss curve consistently decreases without any noticeable turning points. However, there is a slight drawback in terms of accuracy, as we can observe some degree of overfitting.

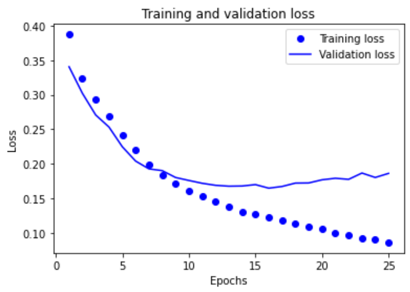
The AUC (Area Under the Curve) is 0.92, indicating high accuracy in the test results. Despite the overfitting concern, the ROC (Receiver Operating Characteristic) curve shows favorable performance. It's important to note that achieving the lowest loss value signifies improved model performance, and the challenge lies in mitigating overfitting while maintaining this trend.

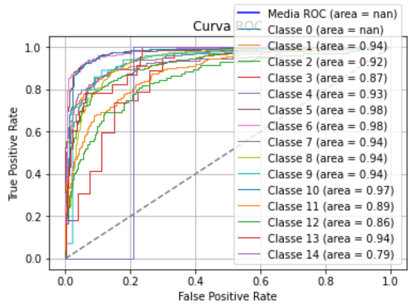
6.4 Experiment 4: two dense layer with 256 neurons

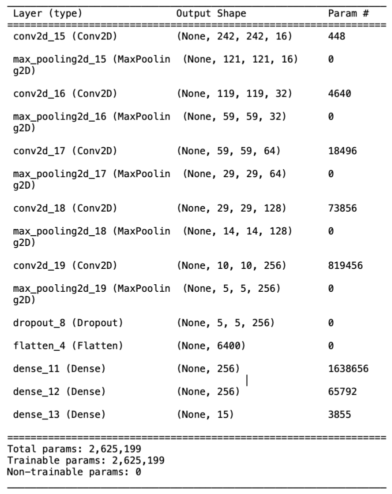
Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.0862	0.8729	0.1862	0.8927

Table 4: Training and validation statistics








```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_15 (Conv2D)          (None, 242, 242, 16)      448

 max_pooling2d_15 (MaxPoolin  (None, 121, 121, 16)     0
 g2D)

 conv2d_16 (Conv2D)          (None, 119, 119, 32)      4640

 max_pooling2d_16 (MaxPoolin  (None, 59, 59, 32)       0
 g2D)

 conv2d_17 (Conv2D)          (None, 59, 59, 64)        18496

 max_pooling2d_17 (MaxPoolin  (None, 29, 29, 64)       0
 g2D)

 conv2d_18 (Conv2D)          (None, 29, 29, 128)       73856

 max_pooling2d_18 (MaxPoolin  (None, 14, 14, 128)      0
 g2D)

 conv2d_19 (Conv2D)          (None, 10, 10, 256)       819456

 max_pooling2d_19 (MaxPoolin  (None, 5, 5, 256)        0
 g2D)

 dropout_8 (Dropout)         (None, 5, 5, 256)         0

 flatten_4 (Flatten)         (None, 6400)              0

 dense_11 (Dense)            (None, 256)               1638656

 dense_12 (Dense)            (None, 256)               65792

 dense_13 (Dense)            (None, 15)                3855

=================================================================
Total params: 2,625,199
Trainable params: 2,625,199
Non-trainable params: 0
_________________________________________________________________
```

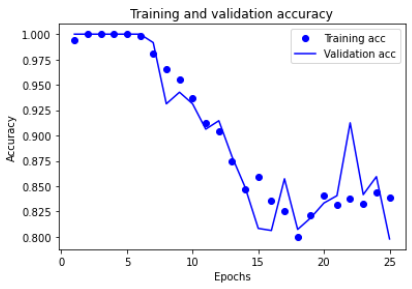
This approach did not lead to significant improvements. We experimented with various combinations of the number of neurons in the two dense layers, such as 128 neurons in the first layer and 256 in the second, but the results did not show notable enhancements.

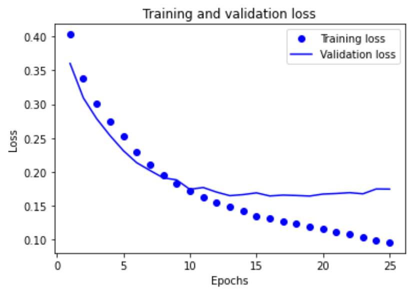
One notable observation is the onset of overfitting, which occurs around epoch 10, much earlier than the previous model, which typically began to exhibit overfitting around epoch 20. This early overfitting is evident in the accuracy metric. While the ROC (Receiver Operating Characteristic) curve shows decent performance, addressing overfitting remains a critical challenge in this scenario.

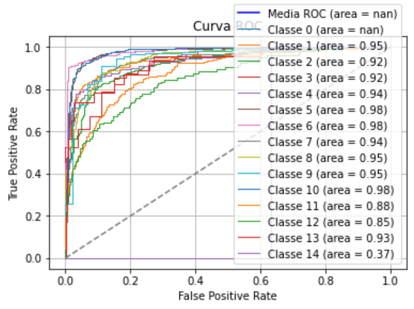
6.5 Experiment 5: two dense layer with 128 and 256 neurons

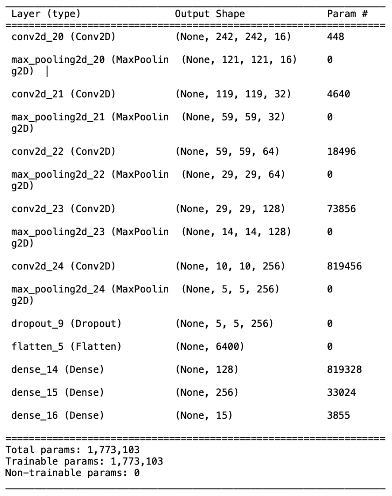
Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.0957	0.8385	0.1744	0.7979

Table 5: Training and validation statistics








```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_20 (Conv2D)          (None, 242, 242, 16)      448

 max_pooling2d_20 (MaxPoolin  (None, 121, 121, 16)     0
 g2D)  |

 conv2d_21 (Conv2D)          (None, 119, 119, 32)      4640

 max_pooling2d_21 (MaxPoolin  (None, 59, 59, 32)       0
 g2D)

 conv2d_22 (Conv2D)          (None, 59, 59, 64)        18496

 max_pooling2d_22 (MaxPoolin  (None, 29, 29, 64)       0
 g2D)

 conv2d_23 (Conv2D)          (None, 29, 29, 128)       73856

 max_pooling2d_23 (MaxPoolin  (None, 14, 14, 128)      0
 g2D)

 conv2d_24 (Conv2D)          (None, 10, 10, 256)       819456

 max_pooling2d_24 (MaxPoolin  (None, 5, 5, 256)        0
 g2D)

 dropout_9 (Dropout)         (None, 5, 5, 256)         0

 flatten_5 (Flatten)         (None, 6400)              0

 dense_14 (Dense)            (None, 128)               819328

 dense_15 (Dense)            (None, 256)               33024

 dense_16 (Dense)            (None, 15)                3855

=================================================================
Total params: 1,773,103
Trainable params: 1,773,103
Non-trainable params: 0
_________________________________________________________________
```

Regarding the validation loss, it can be observed that we did not achieve better results compared to the previous attempts. Interestingly, the model does not appear to yield improvements even when employing a configuration with 128 neurons in the first layer and 256 neurons in the second layer, which is the reverse of the previous approach. One consistent observation is the presence of overfitting, which becomes evident from around epoch 10. This overfitting trend is also reflected in the accuracy graph, while the ROC (Receiver Operating Characteristic) curve suggests decent performance.

7 Introduction to Pre-trained Models

Pre-trained models are versatile tools in machine learning with several important functions. They excel at extracting meaningful features from raw data, such as images or text, and represent them in a structured and compact form, which is ideal for downstream machine learning tasks. Pre-trained models can also be fine-tuned on specific datasets or tasks, leveraging their pre-existing knowledge to enhance performance on new, related tasks, even with limited training data. Moreover, they significantly reduce the time and resources required for training deep neural networks from scratch. Pre-trained models often achieve top-tier performance on various benchmark datasets, making them an attractive choice for researchers and practitioners aiming to efficiently address real-world problems.

8 VGG16 and ResNet: Pre-trained Models for Fine-Tuning

Two prominent examples of pre-trained models are VGG16 and ResNet (Residual Networks). These models have gained popularity for their architecture design and performance in image-related tasks:

8.1 VGG16 (Visual Geometry Group 16)

VGG16 is a deep convolutional neural network that was developed by the Visual Geometry Group at the University of Oxford. It is characterized by its simplicity and uniform architecture, consisting of 16 weight layers, including 13 convolutional layers and 3 fully connected layers. VGG16 excels at feature extraction and image classification tasks. When fine-tuned on specific datasets, it can adapt to various computer vision tasks such as object detection, image segmentation, and more.

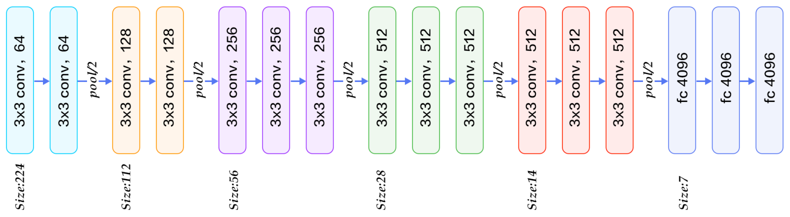


Figure 3: VGG16 architecture

8.2 ResNet (Residual Networks)

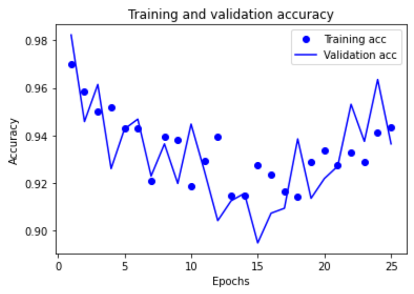
ResNet is a breakthrough architecture that introduced the concept of residual connections or skip connections. These connections allow the network to skip one or more layers, mitigating the vanishing gradient problem and enabling the training of very deep networks. ResNet models, such as ResNet50 and ResNet101, have set performance records in various image recognition competitions, making them a popular choice for transfer learning and fine-tuning on custom datasets.

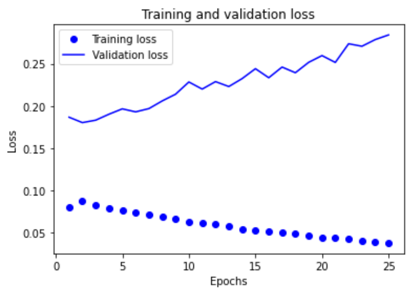
9 VGG16 Experiments

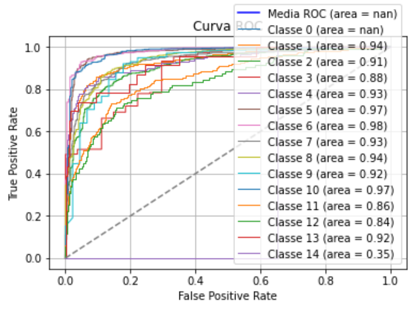
9.1 Experiment 1: adding a dense layer with 256 neurons

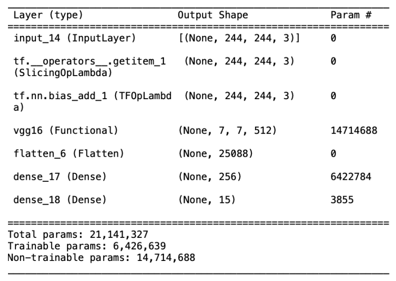
Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.0379	0.9435	0.2841	0.9365

Table 6: Training and validation statistics








```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_14 (InputLayer)       [(None, 244, 244, 3)]     0

 tf.__operators__.getitem_1   (None, 244, 244, 3)      0
 (SlicingOpLambda)

 tf.nn.bias_add_1 (TFOpLambd  (None, 244, 244, 3)      0
 a)

 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten_6 (Flatten)         (None, 25088)             0

 dense_17 (Dense)            (None, 256)               6422784

 dense_18 (Dense)            (None, 15)                3855

=================================================================
Total params: 21,141,327
Trainable params: 6,426,639
Non-trainable params: 14,714,688
_________________________________________________________________
```

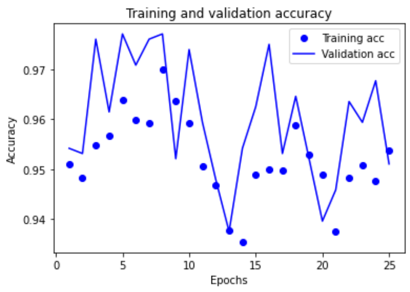
In the scenario where the training loss curve remains consistently low throughout all epochs, indicating that the model is learning well from the training data, while the validation loss curve consistently remains high, hovering around 0.9, it suggests that the model might be in a state of overfitting. This is reflected in the high validation loss curve, as the model struggles to generalize effectively to unseen data.

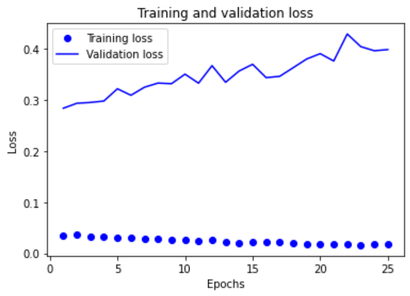
The fact that the training loss curve exhibits "zig-zag" behavior could indicate that the model is trying to adapt too closely to the training data, making frequent weight adjustments during training. This unstable behavior can be an additional signal of overfitting.

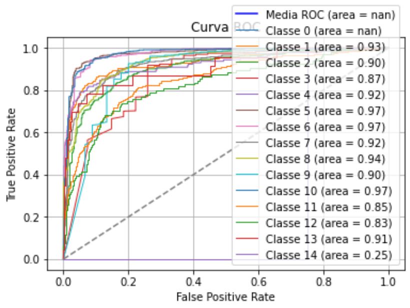
9.2 Experiment 2: one dense layer with 256 neurons and a dropout

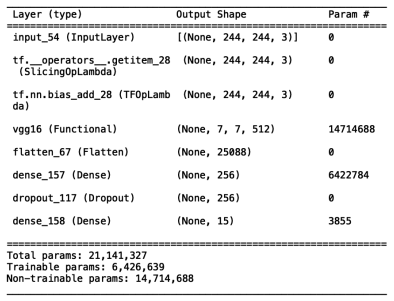
Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.0288	0.9536	0.3985	0.9510

Table 7: Training and validation statistics








```
Layer (type)                Output Shape              Param #
=================================================================
input_54 (InputLayer)       [(None, 244, 244, 3)]     0

tf.__operators__.getitem_28  (None, 244, 244, 3)      0
 (SlicingOpLambda)

tf.nn.bias_add_28 (TFOpLamb  (None, 244, 244, 3)      0
da)

vgg16 (Functional)          (None, 7, 7, 512)         14714688

flatten_67 (Flatten)        (None, 25088)             0

dense_157 (Dense)           (None, 256)               6422784

dropout_117 (Dropout)       (None, 256)               0

dense_158 (Dense)           (None, 15)                3855

=================================================================
Total params: 21,141,327
Trainable params: 6,426,639
Non-trainable params: 14,714,688
_________________________________________________________________
```

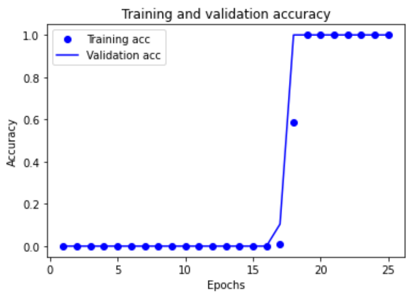
Again, training loss curve remains consistently low, as if the model is learning exceptionally well from the training data, while the validation loss curve consistently stays high, hovering near 0.9, we encounter a situation known as overfitting.

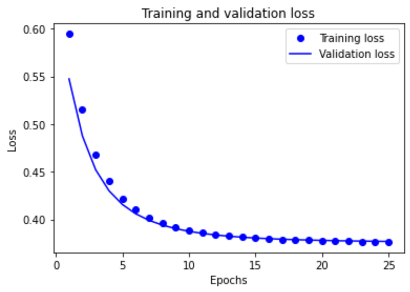
As a result, when we attempt to employ the model on new validation data, it fails to generalize effectively because it has become overly fixated on the specific nuances of the training data, leading to significant errors.

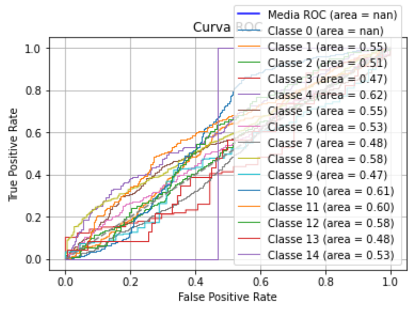
9.3 Experiment 3: GlobalAveragePooling2D

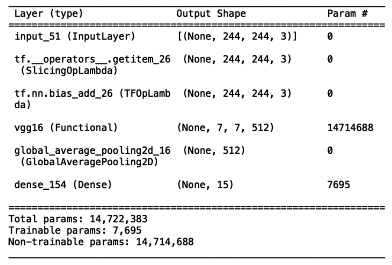
Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.0379	0.9435	0.2841	0.9365

Table 8: Training and validation statistics









```
Layer (type)                Output Shape              Param #
=================================================================
 input_51 (InputLayer)       [(None, 244, 244, 3)]     0

 tf.__operators__.getitem_26  (None, 244, 244, 3)      0
  (SlicingOpLambda)

 tf.nn.bias_add_26 (TFOpLamb  (None, 244, 244, 3)      0
 da)

 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 global_average_pooling2d_16  (None, 512)              0
  (GlobalAveragePooling2D)

 dense_154 (Dense)           (None, 15)                7695

=================================================================
Total params: 14,722,383
Trainable params: 7,695
Non-trainable params: 14,714,688
_________________________________________________________________
```

The training loss curve closely follows the validation loss curve for all the epochs, it indicates that the model is learning effectively from both the training and validation data without showing clear signs of overfitting. This is a positive signal as it suggests that the model can generalize well from the training data to unseen validation data.

However, if the training accuracy curve follows the validation accuracy curve until epoch 17 and then suddenly jumps to 0.90 at epoch 6, it may indicate a situation where the model initially learns very little or nothing from the training data during the first 5 epochs and then suddenly starts to significantly improve its learning capacity.

This suggests that if the model's weights were initialized randomly or with values that make learning difficult, the model may initially have poor performance. Subsequently, during training, the weights are updated through backpropagation, which can lead to a rapid improvement in performance.

In summary, when the training loss closely aligns with the validation loss throughout training, it signifies effective learning and generalization. However, abrupt changes in training accuracy may be attributed to initial weight initialization, which can be improved over time through training.

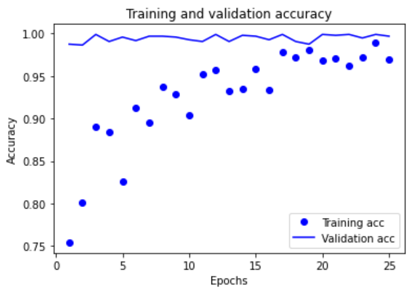
10 Fine Tuning

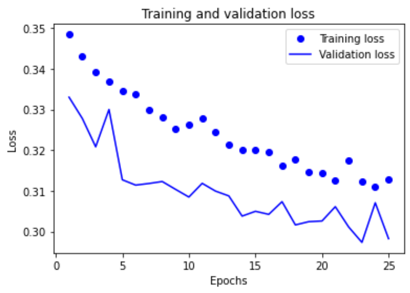
10.1 Experiment 1: last layer dropout model

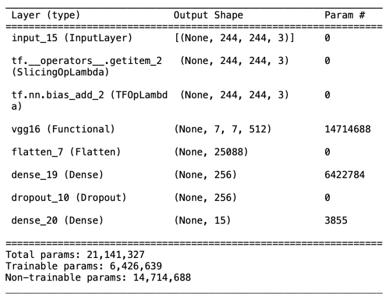
As already anticipated, after running several experiments with the whole convolutional base, we decided to remove block number 5. Again, we performed several experiments in both feature extraction and fine-tuning.

Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.3517	0.6500	0.3265	0.9885

Table 9: Training and validation statistics







```
 Layer (type)                Output Shape              Param #
=================================================================
 input_15 (InputLayer)       [(None, 244, 244, 3)]     0

 tf.__operators__.getitem_2   (None, 244, 244, 3)      0
 (SlicingOpLambda)

 tf.nn.bias_add_2 (TFOpLambd  (None, 244, 244, 3)      0
 a)

 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten_7 (Flatten)         (None, 25088)             0

 dense_19 (Dense)            (None, 256)               6422784

 dropout_10 (Dropout)        (None, 256)               0

 dense_20 (Dense)            (None, 15)                3855

=================================================================
Total params: 21,141,327
Trainable params: 6,426,639
Non-trainable params: 14,714,688
_________________________________________________________________
```

If the training loss curve closely follows the validation loss curve in a decreasing manner towards zero as epochs increase, while the validation accuracy curve remains consistently high at 1, and the training accuracy curve gradually approaches the validation accuracy curve, it signifies that the model is effectively learning from the training data and generalizing well to the validation data.

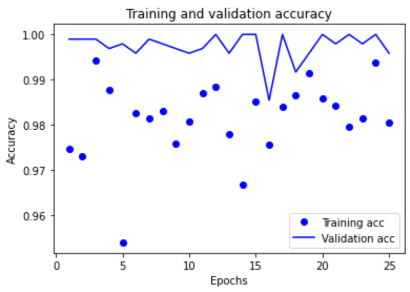
In essence, the decreasing training loss indicates that the model is reducing errors on the training data, and the similar trend in the validation loss indicates it's doing the same on validation data. This is a positive sign, suggesting that the model is learning from both sets of data without exhibiting overfitting symptoms.

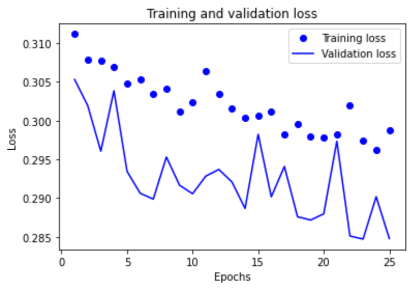
The gradual increase of the training accuracy curve towards the validation accuracy curve suggests that the model is improving its performance on the training data over epochs, but it does so without overfitting, as the two curves (training and validation) remain close to each other.

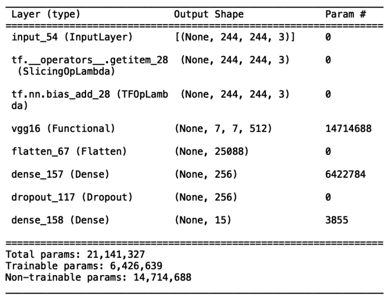
10.2 Experiment 2: last block

Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.2988	0.9805	0.2848	0.9921

Table 10: Training and validation statistics






```
 Layer (type)                Output Shape              Param #
=================================================================
 input_54 (InputLayer)       [(None, 244, 244, 3)]     0

 tf.__operators__.getitem_28  (None, 244, 244, 3)      0
  (SlicingOpLambda)

 tf.nn.bias_add_28 (TFOpLamb  (None, 244, 244, 3)      0
 da)

 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten_67 (Flatten)        (None, 25088)             0

 dense_157 (Dense)           (None, 256)               6422784

 dropout_117 (Dropout)       (None, 256)               0

 dense_158 (Dense)           (None, 15)                3855

=================================================================
Total params: 21,141,327
Trainable params: 6,426,639
Non-trainable params: 14,714,688
_________________________________________________________________
```

If the training loss curve follows the validation loss curve decreasing towards zero as the epochs progress but consistently remains below it with zigzags, while the validation accuracy curve stays high at 1 and the training accuracy curve consists of scattered points, this could suggest a situation of overfitting.

The high validation accuracy at 1 indicates that the model is making correct predictions on all validation data, but the lack of progress in the training accuracy may suggest that the model is struggling to learn from the training data.

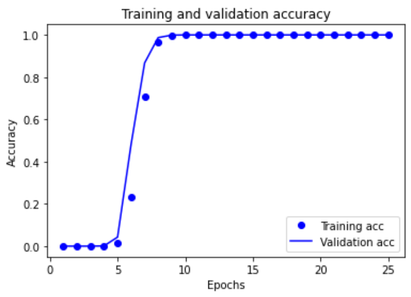
The presence of scattered points in the training accuracy curve could indicate that the model is having difficulty converging towards a stable solution and is oscillating between different states during training.

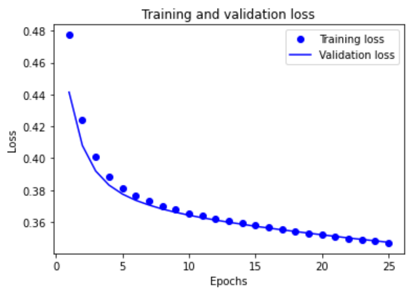
In summary, this situation may suggest overfitting, where the model appears to focus too much on the details of the training data without generalizing well to the validation data.

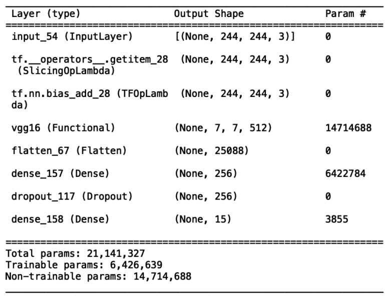
10.3 Experiment 3: last two blocks

Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.3469	0.99	0.3472	0.9921

Table 11: Training and validation statistics







```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_54 (InputLayer)       [(None, 244, 244, 3)]     0

 tf.__operators__.getitem_28  (None, 244, 244, 3)      0
  (SlicingOpLambda)

 tf.nn.bias_add_28 (TFOpLamb  (None, 244, 244, 3)      0
 da)

 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten_67 (Flatten)        (None, 25088)             0

 dense_157 (Dense)           (None, 256)               6422784

 dropout_117 (Dropout)       (None, 256)               0

 dense_158 (Dense)           (None, 15)                3855

=================================================================
Total params: 21,141,327
Trainable params: 6,426,639
Non-trainable params: 14,714,688
_________________________________________________________________
```

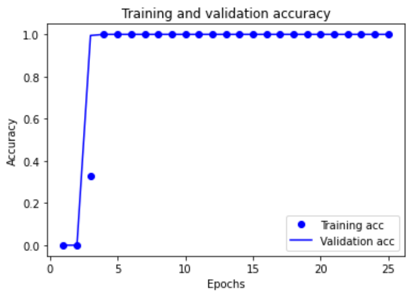
The training loss curve closely follows the validation loss curve for all the epochs, it indicates that the model is learning effectively from both the training and validation data without showing clear signs of overfitting. This is a positive signal as it suggests that the model can generalize well from the training data to unseen validation data.

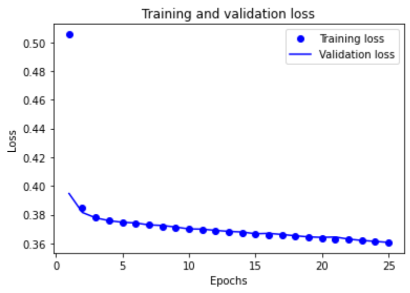
11 ResNet Experiments

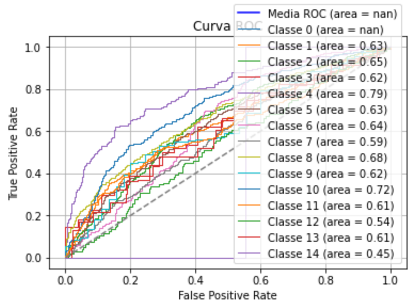
11.1 Experiment 1: one GlobalAveragePooling layer

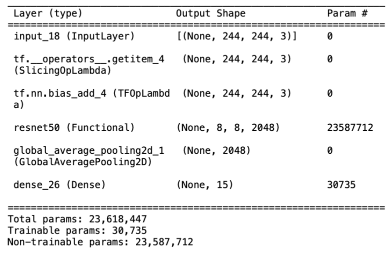
Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.3616	0.989	0.3614	0.9921

Table 12: Training and validation statistics









```
 Layer (type)                Output Shape              Param #
=================================================================
 input_18 (InputLayer)       [(None, 244, 244, 3)]     0

 tf.__operators__.getitem_4   (None, 244, 244, 3)      0
 (SlicingOpLambda)

 tf.nn.bias_add_4 (TFOpLambd  (None, 244, 244, 3)      0
 a)

 resnet50 (Functional)       (None, 8, 8, 2048)        23587712

 global_average_pooling2d_1   (None, 2048)             0
 (GlobalAveragePooling2D)

 dense_26 (Dense)            (None, 15)                30735

=================================================================
Total params: 23,618,447
Trainable params: 30,735
Non-trainable params: 23,587,712
```

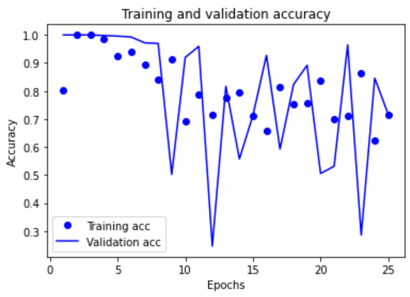
In this situation, it appears that the model has perfectly memorized the training data from the very first epoch, achieving a training accuracy of 1. However, since the validation accuracy curve consistently remains high at 1, it could indicate that the model is not generalizing well to the validation data and is behaving more like a memorizer than a learner.

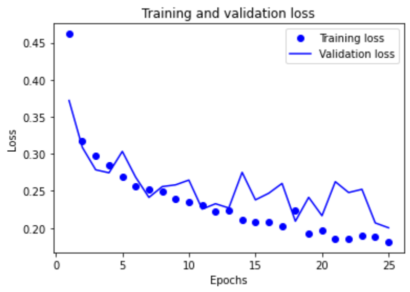
This situation is indicative of extreme overfitting, where the model is highly tailored to the training data but struggles to generalize significantly.

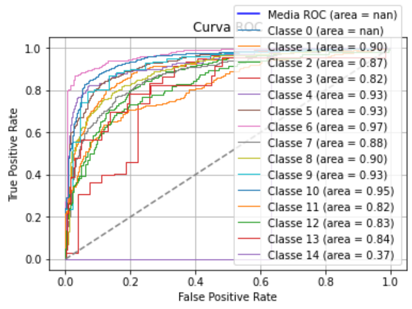
11.2 Experiment 2: one dense layer with 64 neurons

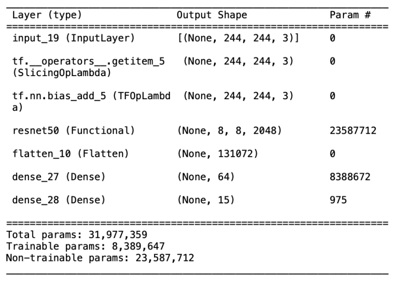
Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.1852	0.7096	0.2478	0.9646

Table 13: Training and validation statistics









```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_19 (InputLayer)       [(None, 244, 244, 3)]     0

 tf.__operators__.getitem_5   (None, 244, 244, 3)      0
 (SlicingOpLambda)

 tf.nn.bias_add_5 (TFOpLambd  (None, 244, 244, 3)      0
 a)

 resnet50 (Functional)       (None, 8, 8, 2048)        23587712

 flatten_10 (Flatten)        (None, 131072)            0

 dense_27 (Dense)            (None, 64)                8388672

 dense_28 (Dense)            (None, 15)                975

=================================================================
Total params: 31,977,359
Trainable params: 8,389,647
Non-trainable params: 23,587,712
_________________________________________________________________
```

If the training loss curve follows the validation loss curve in a decreasing manner towards zero as the epochs increase, but starts from a medium-low value, while the validation accuracy curve begins high and shows some fluctuations in the middle of the graph with the training curve following it, it may indicate a situation where the model is effectively learning from the training data but is having difficulty generalizing stably to the validation data.

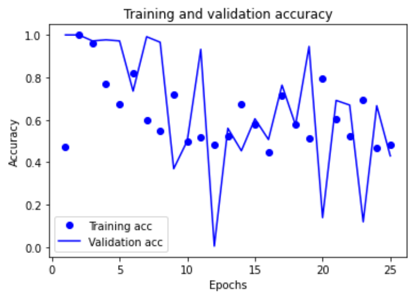
The medium-low starting point of the training loss curve suggests that the model begins with reasonable performance on the training data. However, the fluctuations in the validation accuracy curve indicate that the model is facing some challenges in generalizing to the validation data.

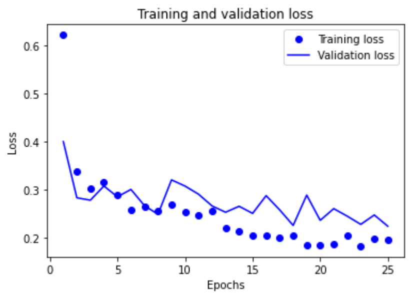
This scenario suggests that the model is learning well from the training data but faces challenges in consistent generalization to the validation data.

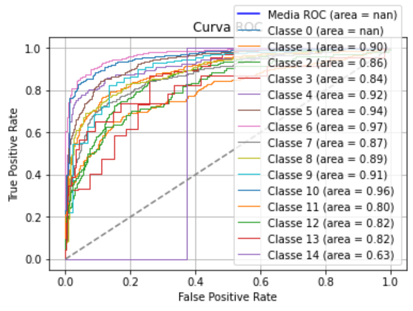
11.3 Experiment 3: one dense layer with 256 neurons

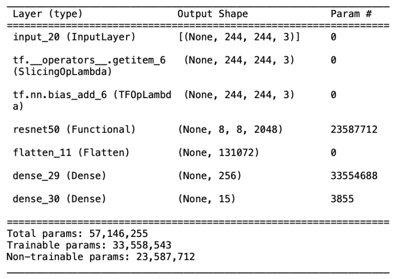
Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.1975	0.4805	0.2249	0.4302

Table 14: Training and validation statistics







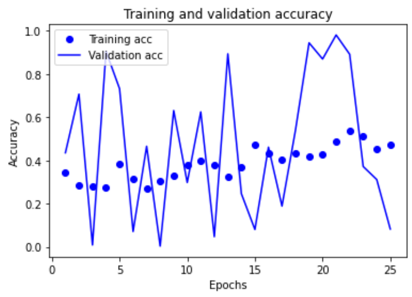

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_20 (InputLayer)       [(None, 244, 244, 3)]     0

 tf.__operators__.getitem_6   (None, 244, 244, 3)      0
 (SlicingOpLambda)

 tf.nn.bias_add_6 (TFOpLambd  (None, 244, 244, 3)      0
 a)

 resnet50 (Functional)       (None, 8, 8, 2048)        23587712

 flatten_11 (Flatten)        (None, 131072)            0

 dense_29 (Dense)            (None, 256)               33554688

 dense_30 (Dense)            (None, 15)                3855

=================================================================
Total params: 57,146,255
Trainable params: 33,558,543
Non-trainable params: 23,587,712
_________________________________________________________________
```

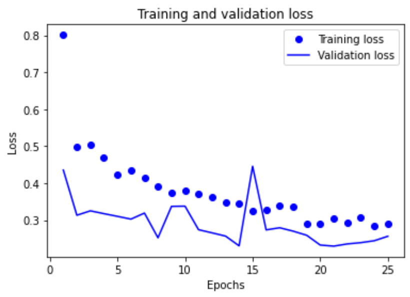
This scenario is the same as before, suggests that the model is learning well from the training data but faces challenges in consistent generalization to the validation data.

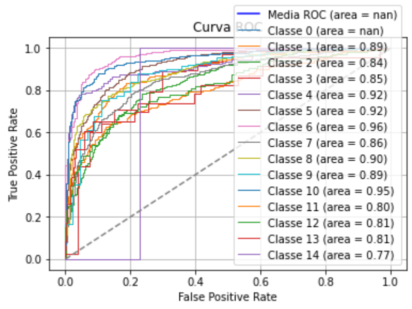
11.4 Experiment 4: one dense layer with 256 neurons and a dropout

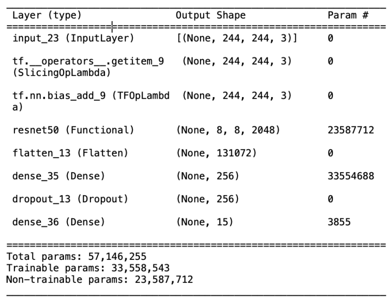
Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.2893	0.4745	0.2565	0.0823

Table 15: Training and validation statistics







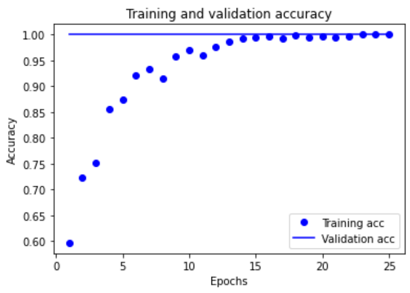

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_23 (InputLayer)       [(None, 244, 244, 3)]     0

 tf.__operators__.getitem_9   (None, 244, 244, 3)      0
 (SlicingOpLambda)

 tf.nn.bias_add_9 (TFOpLambd  (None, 244, 244, 3)      0
 a)

 resnet50 (Functional)       (None, 8, 8, 2048)        23587712

 flatten_13 (Flatten)        (None, 131072)            0

 dense_35 (Dense)            (None, 256)               33554688

 dropout_13 (Dropout)        (None, 256)               0

 dense_36 (Dense)            (None, 15)                3855

=================================================================
Total params: 57,146,255
Trainable params: 33,558,543
Non-trainable params: 23,587,712
_________________________________________________________________
```

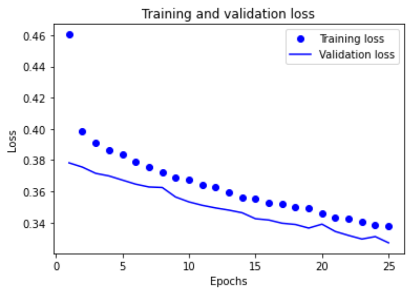
The fluctuations in the validation accuracy curve indicate that the model is facing some challenges in generalizing to the validation data. These fluctuations can be caused by various factors, such as a small validation dataset size or high model complexity.

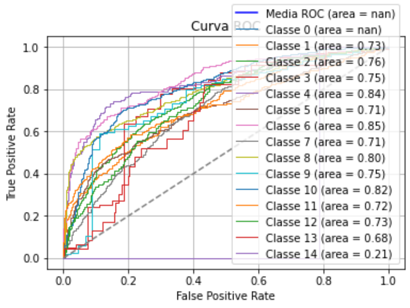
11.5 Experiment 5: one dense layer, one GlobalAveragePooling2D and a dropout

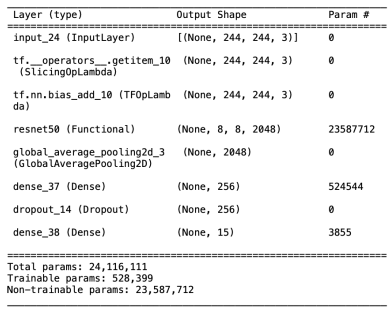
Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.3403	0.9995	0.3295	0.998

Table 16: Training and validation statistics








```
 Layer (type)                Output Shape              Param #
=================================================================
 input_24 (InputLayer)       [(None, 244, 244, 3)]     0

 tf.__operators__.getitem_10  (None, 244, 244, 3)      0
  (SlicingOpLambda)

 tf.nn.bias_add_10 (TFOpLamb  (None, 244, 244, 3)      0
 da)

 resnet50 (Functional)       (None, 8, 8, 2048)        23587712

 global_average_pooling2d_3   (None, 2048)             0
 (GlobalAveragePooling2D)

 dense_37 (Dense)            (None, 256)               524544

 dropout_14 (Dropout)        (None, 256)               0

 dense_38 (Dense)            (None, 15)                3855

=================================================================
Total params: 24,116,111
Trainable params: 528,399
Non-trainable params: 23,587,712
_________________________________________________________________
```

The fact that the training loss approaches zero suggests that the model is reducing the error on the training data, and the same holds true for the validation loss, which is a positive sign and suggests that the model is learning effectively from both datasets without clear signs of overfitting.

The consistently high validation accuracy indicates that the model is making correct predictions on all validation data, which is a very positive outcome.

The gradual increase in training accuracy until it matches the validation accuracy suggests that the model is improving its performance on the training data over the epochs without overfitting, as the two curves (training and validation) remain close to each other.

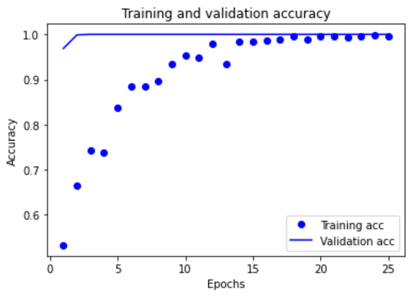
12 Fine Tuning

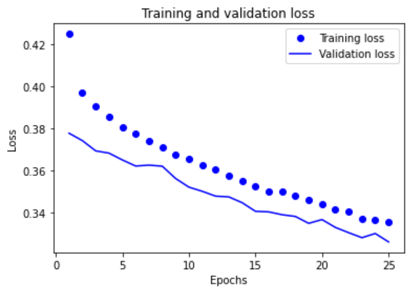
12.1 Experiment 1: last layer dense layer 64 neurons

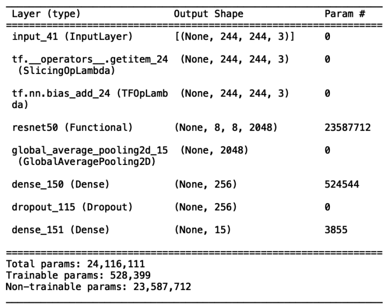
It has a graph similar to the previous one, so the analogous considerations apply.

Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.2996	0.9890	0.3156	0.985

Table 17: Training and validation statistics







```
Layer (type)                Output Shape              Param #
=================================================================
 input_41 (InputLayer)       [(None, 244, 244, 3)]     0

 tf.__operators__.getitem_24  (None, 244, 244, 3)      0
  (SlicingOpLambda)

 tf.nn.bias_add_24 (TFOpLamb  (None, 244, 244, 3)      0
 da)

 resnet50 (Functional)       (None, 8, 8, 2048)        23587712

 global_average_pooling2d_15  (None, 2048)             0
  (GlobalAveragePooling2D)

 dense_150 (Dense)           (None, 256)               524544

 dropout_115 (Dropout)       (None, 256)               0

 dense_151 (Dense)           (None, 15)                3855

=================================================================
Total params: 24,116,111
Trainable params: 528,399
Non-trainable params: 23,587,712
_________________________________________________________________
```

The high validation accuracy at 1 indicates that the model is making correct predictions on all validation data, but the lack of progress in the training accuracy may suggest that the model is struggling to learn from the training data.

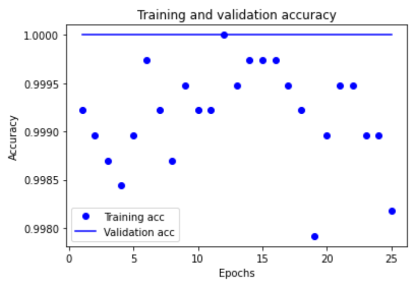
The presence of scattered points in the training accuracy curve could indicate that the model is having difficulty converging towards a stable solution and is oscillating between different states during training, so we have overfitting.

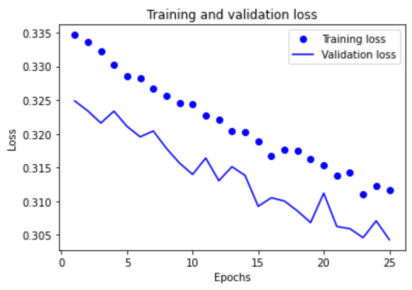
12.2 Experiment 2: last layer one dense layer with 256 neurons

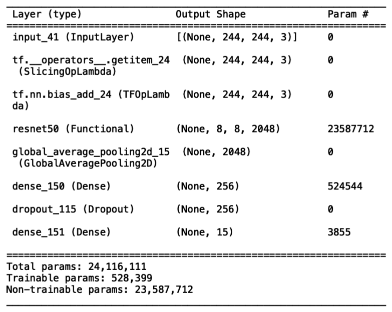
It has a graph similar to the previous one, so the analogous considerations apply.

Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.3205	0.99	0.3345	0.996

Table 18: Training and validation statistics







```
 Layer (type)                Output Shape              Param #
=================================================================
 input_41 (InputLayer)       [(None, 244, 244, 3)]     0

 tf.__operators__.getitem_24  (None, 244, 244, 3)      0
  (SlicingOpLambda)

 tf.nn.bias_add_24 (TFOpLamb  (None, 244, 244, 3)      0
 da)

 resnet50 (Functional)       (None, 8, 8, 2048)        23587712

 global_average_pooling2d_15  (None, 2048)             0
  (GlobalAveragePooling2D)

 dense_150 (Dense)           (None, 256)               524544

 dropout_115 (Dropout)       (None, 256)               0

 dense_151 (Dense)           (None, 15)                3855

=================================================================
Total params: 24,116,111
Trainable params: 528,399
Non-trainable params: 23,587,712
_________________________________________________________________
```

The presence of scattered points in the training accuracy curve could indicate that the model is having difficulty converging towards a stable solution and is oscillating between different states during training.

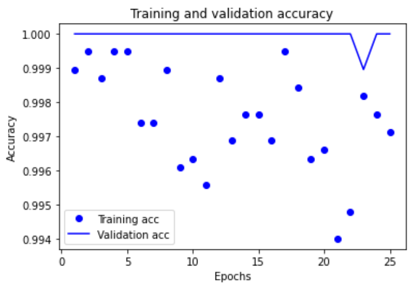
In summary, this situation may suggest overfitting, where the model appears to focus too much on the details of the training data without generalizing well to the validation data.

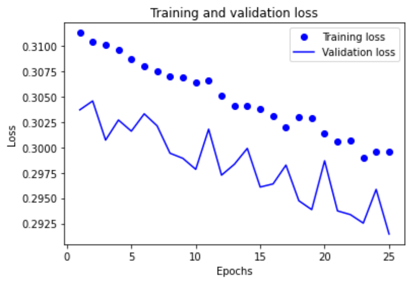
12.3 Experiment 3: last layer one dense layer 256 neurons

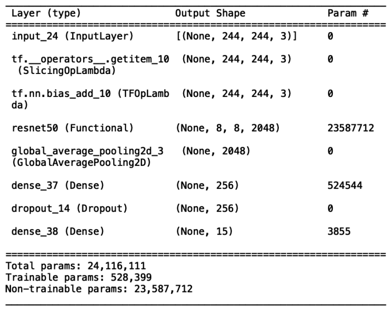
It has a graph similar to the previous one, so the analogous considerations apply.

Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.2995	0.9971	0.2915	0.999

Table 19: Training and validation statistics







```
________________________________________________________________
 Layer (type)                Output Shape              Param #
================================================================
 input_24 (InputLayer)       [(None, 244, 244, 3)]     0

 tf.__operators__.getitem_10  (None, 244, 244, 3)      0
  (SlicingOpLambda)

 tf.nn.bias_add_10 (TFOpLamb  (None, 244, 244, 3)      0
 da)

 resnet50 (Functional)       (None, 8, 8, 2048)        23587712

 global_average_pooling2d_3   (None, 2048)             0
 (GlobalAveragePooling2D)

 dense_37 (Dense)            (None, 256)               524544

 dropout_14 (Dropout)        (None, 256)               0

 dense_38 (Dense)            (None, 15)                3855

================================================================
Total params: 24,116,111
Trainable params: 528,399
Non-trainable params: 23,587,712
________________________________________________________________
```

Overfitting is a critical challenge because our ultimate objective is to have models that can reliably generalize to new real-world data. Therefore, we must find ways to address overfitting, such as utilizing techniques like regularization, reducing model complexity, or increasing the size of the training dataset.

In summary, when we observe a persistently low training loss curve accompanied by a consistently high validation loss curve, it signals that our model is suffering from overfitting, and it is crucial to tackle this issue to enhance the model's real-world performance

13 Bibliography

[1] **"Mask R-CNN"**
Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick
Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017

[2] **"Panoptic Feature Pyramid Networks"**
Alexander Kirillov, Ross Girshick, Kaiming He, Piotr Dollár
Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019

[3] **"Fully Convolutional Networks for Semantic Segmentation"**
Jonathan Long, Evan Shelhamer, Trevor Darrell
Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015

[4] **"U-Net: Convolutional Networks for Biomedical Image Segmentation"**
Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2015
CVPR, 2019

[5] **"DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs"**
Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, Alan L. Yuille
IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2018

[6] **"Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation"**
Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, Hartwig Adam
Proceedings of the European Conference on Computer Vision (ECCV), 2018

Category	Class	Class ID	Train ID	ROC Class	Description
Others	others	0	255		
	rover	1	255		
Sky	sky	17	0		
Movable Object	car	33	1	0	
	car_groups	161	1	9	
	motorbicycle	34	2	1	
	motorbicycle_group	162	2	10	
	bicycle	35	3	3	
	bicycle_group	163	3	11	
	person	36	4	4	
	person_group	164	4	12	
	rider	37	5	5	person on motorcycle, bicycle or tricycle
	rider_group	165	5	13	person on motorcycle, bicycle or tricycle
	truck	38	6	6	
	truck_group	166	6	14	
	bus	39	7	7	
	bus_group	167	7	15	
	tricycle	40	8	8	three-wheeled vehicles, motorized, or human-powered
	tricycle_group	168	8	16	three-wheeled vehicles, motorized, or human-powered
Flat	road	49	9		
	sidewalk	50	10		
Road Obstacles	traffic_cone	65	11		movable and cone-shaped markers
	road_pile	66	12		fixed with many different shapes
	fence	67	13		
Roadside Objects	traffic_light	81	14		
Void	pole	82	15		
	traffic_sign	83	16		
	wall	84	17		
	dustbin	85	18		
	billboard	86	19		
Building	building	97	20		
	bridge	98	255		
	tunnel	99	255		
	overpass	100	255		
Natural	vegetation	113	21		
Unlabeled	unlabeled	255	255		other unlabeled objects

Table 20: Description of Categories and Classes