Deep Learning for Computer Vision Assignment

DELLA VALLE FRANCESCO
LALE DELIL
SAVIANO GAETANO

DEEP LEARNING
APRIL 2024

Image Recognition Problem

Image recognition is crucial in computer vision, as it involves recognizing and categorizing objects, scenes, or patterns in digital images or video frames. Its significance resonates across diverse real-world applications, spanning autonomous driving, medical diagnosis, surveillance, and augmented reality.
The heart of image recognition lies in creating algorithms that can understand and interpret the complex visual information contained within images. Traditional methodologies heavily leaned on manually crafted features and *shallow* machine learning models, often grappling with the complexities inherent in visual patterns and the variability introduced by *lighting conditions, viewpoints, and background clutter*.
However, with the emergence of deep learning, particularly **convolutional neural networks (CNNs)**, substantial strides have been made in the domain of image recognition. By traversing multiple layers of convolution, pooling, and non-linear transformations, CNNs adeptly extract increasingly abstract features, thereby facilitating precise recognition of objects and patterns depicted in images.
However, despite these remarkable advancements, image recognition still presents significant challenges due to factors such as occlusions, variations in scale and orientation, and the presence of multiple objects within a single scene

Dataset

The xView dataset emerges as a significant open repository for object detection, featuring an expansive collection of approximately 1 million annotated objects spanning across 60 distinct categories. This dataset comprises meticulously labeled images sourced from various global scenes, captured by the WorldView-3 satellite at a ground sample distance of 0.3 meters. Within this dataset, a total of 846 images are annotated, with 761 allocated for training and 85 for testing purposes.
In our analysis, we streamline the dataset by focusing solely on the 12 object categories. These categories and their respective sample counts are as follows:

- **Cargo plane**: 635
- **Helicopter**: 70
- **Small car:** 4290
- **Bus:** 2155
- **Truck:** 2746
- **Motorboat:** 1069
- **Fishing vessel:** 706
- **Dump truck:** 1236
- **Excavator:** 789
- **Building:** 4689
- **Storage tank:** 1469
- **Shipping container:** 1523

For image recognition tasks, we utilize bounding box annotations to extract a subset of objects of interest from the original images. Through this process, we distribute a total of

21,377 objects for training and **2,635** for testing. Subsequently, these extracted objects are resized to a standardized dimension of **224x224 pixels**, ensuring uniformity across the dataset.

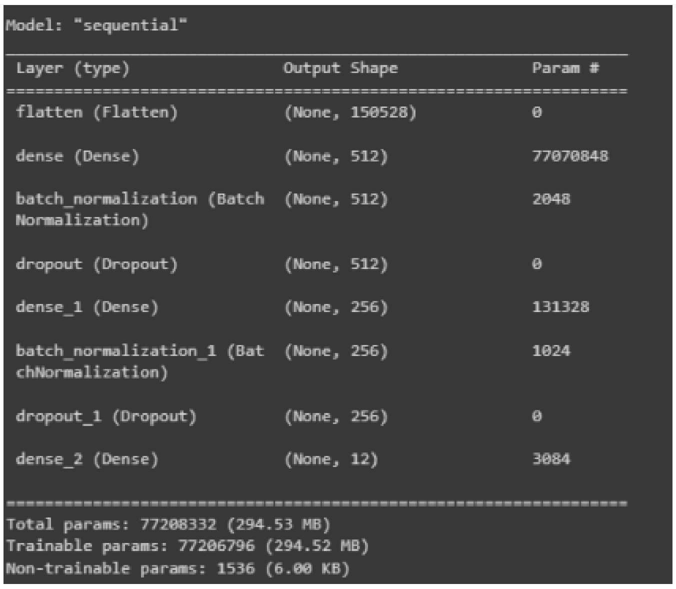
FeedForwarded Network

The initial reference point for our work was the provided *ffNN_example.ipynb*, which outlines a basic neural network architecture constructed using TensorFlow's Keras API. This architecture is structured as a Sequential model comprising a Flatten layer, succeeded by an Activation layer utilizing the ReLU activation function, a Dropout layer with a dropout rate of 0.2, and a Dense layer with units corresponding to the number of categories, capped with a Softmax activation function.

Subsequently, we iteratively refined and enhanced the initial architecture to augment its performance and generalization capabilities. Noteworthy modifications entail the introduction of Batch Normalization layers post the Dense layers and alterations to the configuration of the dense layers. Specifically, the revised architecture incorporates two Dense layers with 512 and 256 units, respectively, both activated by ReLU functions. Additionally, each Dense layer is immediately succeeded by a Batch Normalization layer, aimed at stabilizing and expediting the training process. Furthermore, the dropout rate is elevated to 0.5 for both Dropout layers, potentially furnishing more robust regularization to mitigate overfitting concerns. The concluding Dense layer preserves the Softmax activation function.
These enhancements are directed towards refining the neural network's training stability and augmenting its capacity to discern intricate patterns within the data, thereby fostering improved performance and generalization prowess.

FeedForward Neural Network v1

Below is the description regarding the proposed neural network architecture for the assignment.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 150528)            0

 dense (Dense)               (None, 512)               77070848

 batch_normalization (Batch  (None, 512)               2048
 Normalization)

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 batch_normalization_1 (Bat  (None, 256)               1024
 chNormalization)

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 12)                3084

=================================================================
Total params: 77208332 (294.53 MB)
Trainable params: 77206796 (294.52 MB)
Non-trainable params: 1536 (6.00 KB)
```

The model architecture starts with an input layer that flattens the input images, converting them into a one-dimensional array for further processing. Following this, there are two hidden layers, each consisting of densely connected neurons with ***Rectified Linear Unit (ReLU)*** activation functions. These layers are essential for learning complex representations of image features.
To enhance the training process and prevent overfitting, batch normalization layers are incorporated after each hidden layer. ***Batch normalization*** helps in stabilizing and accelerating the training process by normalizing the activations.
Additionally, ***dropout layers*** are introduced after each batch normalization layer. Dropout randomly drops a certain percentage of neurons during training, effectively reducing the model's dependency on specific neurons and features, thus improving generalization.
The output layer is a densely connected layer with a ***softmax activation function***. This layer produces probabilities for each class in the classification task, enabling the model to make predictions based on the highest probability class.

Model Optimization and Compilation - FeedForward Neural Network v1

For optimization, the *Adam optimizer* is utilized with the following configuration:

- **Learning rate:** 1e-3
- **Beta 1**: 0.9
- **Beta 2:** 0.999
- **Epsilon:** 1e-8
- **AMSGrad:** Enabled
- **Gradient clipping norm:** 1.0
- **Value clipping:** 0.5

The model is compiled using the *categorical crossentropy loss function* and *accuracy* as the evaluation metric.

Batch Size and Epochs - FeedForward Neural Network v1

The training process utilizes a **batch size of 16**, meaning that each iteration during training processes 32 samples simultaneously. The training loop runs for **20 epochs**, which defines

Callbacks for Model Training FeedForward Neural Network v1

During the training process of the neural network model, several callbacks are employed to monitor its progress and make necessary adjustments. These callbacks serve various purposes such as saving the best model, adjusting learning rate dynamically, and terminating training under certain conditions.

ModelCheckpoint: This callback, named model_checkpoint, monitors the validation accuracy (val_accuracy) during training. It saves the model weights to a file named 'model.hdf5' whenever there is an improvement in validation accuracy, ensuring that the best-performing model is retained.

ReduceLROnPlateau: The reduce_lr callback dynamically adjusts the learning rate during training based on the validation accuracy (val_accuracy). If the validation accuracy fails to improve after a certain number of epochs (patience), the learning rate is reduced by a factor of 0.1. This facilitates finer adjustments to the learning process, potentially aiding convergence.

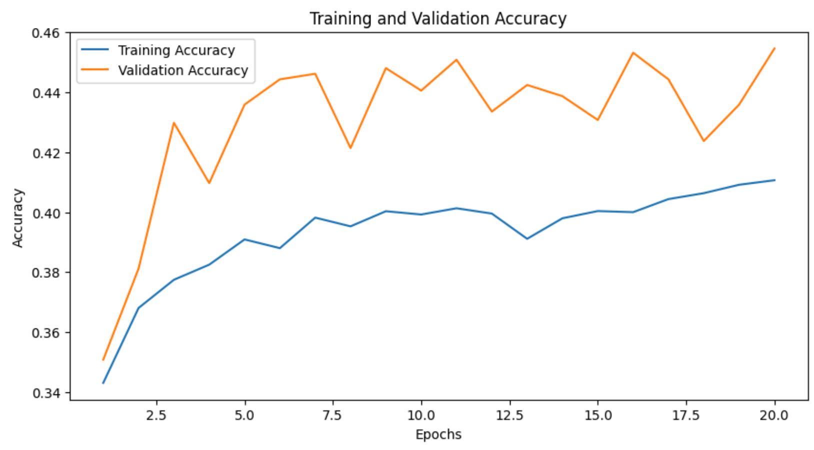
EarlyStopping: Named early_stop, this callback monitors the validation accuracy (val_accuracy) and halts training if there is no improvement observed for a specified number of epochs (patience). In this case, training is stopped after 40 epochs of no improvement, thereby preventing unnecessary computational overhead and potential overfitting.

TerminateOnNaN: This callback, named terminate, ensures that training terminates if any NaN (not a number) values are encountered during training. This is a precautionary measure to prevent invalid calculations or model instability.

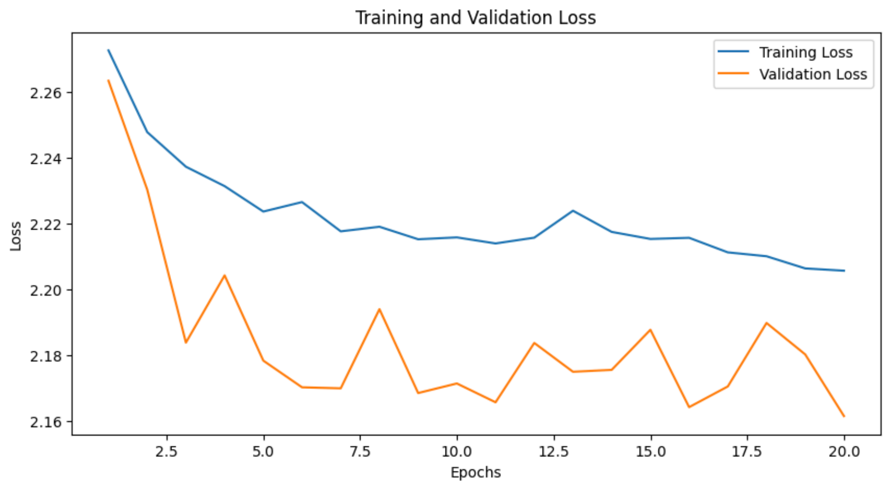
These callbacks are instrumental in enhancing the efficiency and effectiveness of the training process, enabling the model to achieve optimal performance while mitigating the risk of overfitting and instability.

Results - FeedForward Neural Network v1

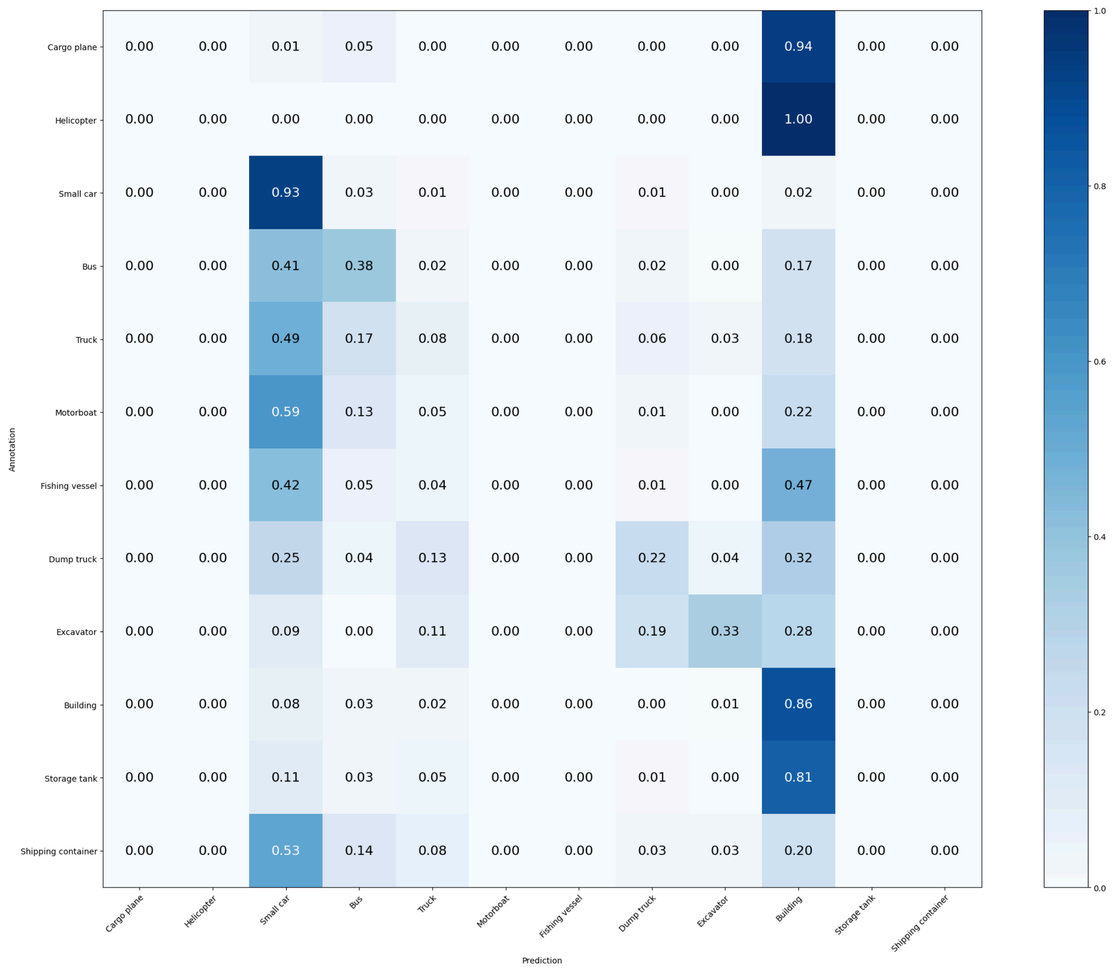
The model achieved its highest validation accuracy of approximately **0.4546304941177368%** at epoch 20, indicating its best performance on unseen data during training.
Training and Validation Accuracy curves show a huge problem of underfitting.



Training and Validation loss curves show a really big instability into the model.



Analyzing the confusion matrix, the model recognizes only “Buildings” and “Small Cars”, for the other classes the results are near to 0.



	Cargo plane	Helicopter	Small car	Bus	Truck	Motorboat	Fishing vessel	Dump truck	Excavator	Building	Storage tank	Shipping container
Cargo plane	0.00	0.00	0.01	0.05	0.00	0.00	0.00	0.00	0.00	0.94	0.00	0.00
Helicopter	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
Small car	0.00	0.00	0.93	0.03	0.01	0.00	0.00	0.01	0.00	0.02	0.00	0.00
Bus	0.00	0.00	0.41	0.38	0.02	0.00	0.00	0.02	0.00	0.17	0.00	0.00
Truck	0.00	0.00	0.49	0.17	0.08	0.00	0.00	0.06	0.03	0.18	0.00	0.00
Motorboat	0.00	0.00	0.59	0.13	0.05	0.00	0.00	0.01	0.00	0.22	0.00	0.00
Fishing vessel	0.00	0.00	0.42	0.05	0.04	0.00	0.00	0.01	0.00	0.47	0.00	0.00
Dump truck	0.00	0.00	0.25	0.04	0.13	0.00	0.00	0.22	0.04	0.32	0.00	0.00
Excavator	0.00	0.00	0.09	0.00	0.11	0.00	0.00	0.19	0.33	0.28	0.00	0.00
Building	0.00	0.00	0.08	0.03	0.02	0.00	0.00	0.00	0.01	0.86	0.00	0.00
Storage tank	0.00	0.00	0.11	0.03	0.05	0.00	0.00	0.01	0.00	0.81	0.00	0.00
Shipping container	0.00	0.00	0.53	0.14	0.08	0.00	0.00	0.03	0.03	0.20	0.00	0.00

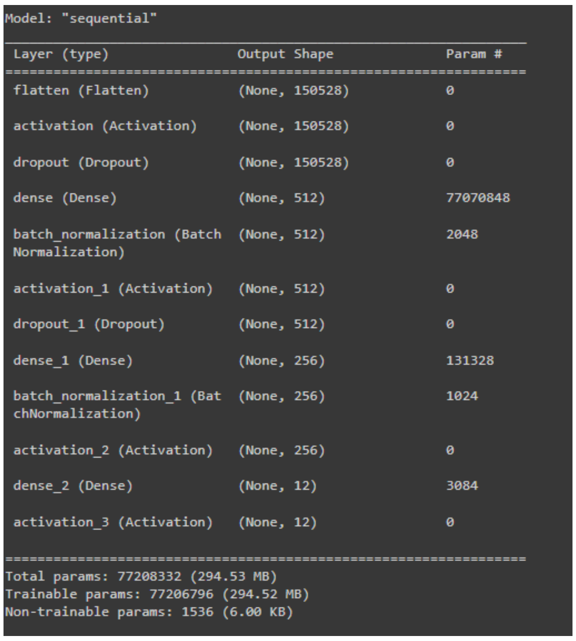
Mean Accuracy: With a mean accuracy of 41.025%, it indicates that, on average, the model correctly classified approximately 41.025% of the instances in the dataset.

Mean Recall: The mean recall of 23.360% suggests that, on average, the model correctly identified approximately 23.360% of the true positive instances among all positive instances in the dataset.

Mean Precision: A mean precision of 19.127% indicates that, on average, approximately 19.127% of the instances predicted as positive by the model were actually true positive instances.

Improved FeedForward Neural Network

The following FF neural network model represents an improved iteration over the architecture initially presented for the optional assignment:



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 150528)            0

 activation (Activation)     (None, 150528)            0

 dropout (Dropout)           (None, 150528)            0

 dense (Dense)               (None, 512)               77070848

 batch_normalization (Batch  (None, 512)               2048
 Normalization)

 activation_1 (Activation)   (None, 512)               0

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 batch_normalization_1 (Bat  (None, 256)               1024
 chNormalization)

 activation_2 (Activation)   (None, 256)               0

 dense_2 (Dense)             (None, 12)                3084

 activation_3 (Activation)   (None, 12)                0

=================================================================
Total params: 77208332 (294.53 MB)
Trainable params: 77206796 (294.52 MB)
Non-trainable params: 1536 (6.00 KB)
```

The primary differences between the two versions of the model are as follows:

- In the original model, the dropout rate was set to 0.5, and the number of units in the dense layers was 512 and 256, respectively. In the modified version, while the dropout rate and the number of units remained the same, adjustments were made to enhance the model's generalization capabilities.

- Furthermore, the modified version incorporates batch normalization layers after each dense layer. This technique was not present in the original version. Batch normalization contributes significantly to stabilizing the training process and accelerating convergence by normalizing the activations of the preceding layers.Regarding the activation function, in the original model, the ReLU activation function was applied after adding the dense and batch normalization layers. However, in the modified version, ReLU activation is applied before adding dropout and batch normalization layers. This alteration in the sequence of operations aims to optimize the model's performance and training dynamics.

In essence, these modifications in the model's architecture and sequence of operations are strategically implemented to refine its performance, stability, and capacity to generalize well to unseen data.

Model Optimization and Compilation - Improved FeedForward Neural Network

For optimization, the *Adam optimizer* is utilized with the following configuration:

- **Learning rate:** 1e-3
- **Beta 1**: 0.9
- **Beta 2:** 0.999
- **Epsilon:** 1e-8
- **AMSGrad:** Enabled
- **Gradient clipping norm:** 1.0
- **Value clipping:** 0.5

The model is compiled using the *categorical crossentropy loss function* and *accuracy* as the evaluation metric.

Batch Size and Epochs - Improved FeedForward Neural Network

The training process utilizes a **batch size of 32**, meaning that each iteration during training processes 32 samples simultaneously. The training loop runs for **20 epochs**, which defines the number of times the entire training dataset is passed forward and backward through the neural network.

Callbacks for Model Training - Improved FeedForward Neural Network

During the training process of the neural network model, several callbacks are employed to monitor its progress and make necessary adjustments. These callbacks serve various purposes such as saving the best model, adjusting learning rate dynamically, and terminating training under certain conditions.

ModelCheckpoint: This callback, named model_checkpoint, monitors the validation accuracy (val_accuracy) during training. It saves the model weights to a file named 'model.hdf5' whenever there is an improvement in validation accuracy, ensuring that the best-performing model is retained.

ReduceLROnPlateau: The reduce_lr callback dynamically adjusts the learning rate during training based on the validation accuracy (val_accuracy). If the validation accuracy fails to improve after a certain number of epochs (patience), the learning rate is reduced by a factor of 0.1. This facilitates finer adjustments to the learning process, potentially aiding convergence.

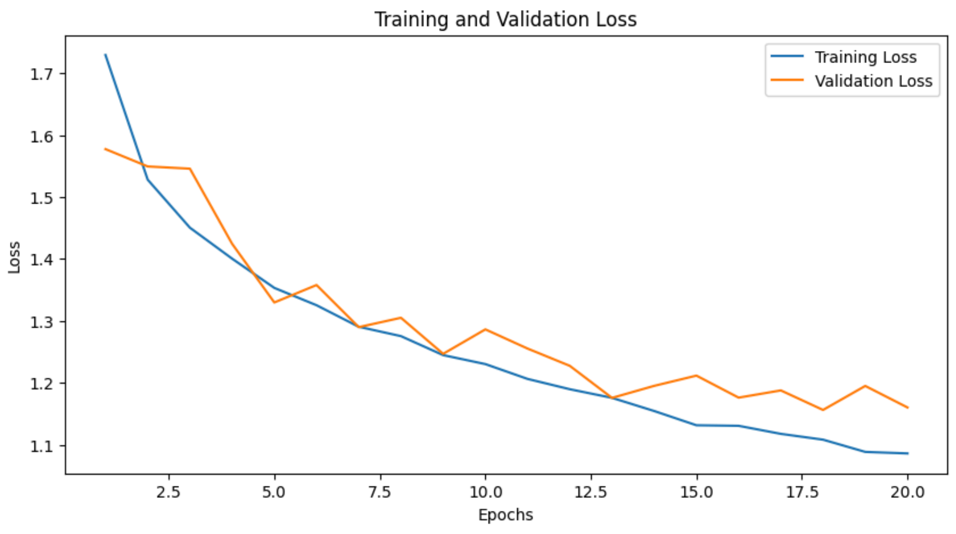
EarlyStopping: Named early_stop, this callback monitors the validation accuracy (val_accuracy) and halts training if there is no improvement observed for a specified number of epochs (patience). In this case, training is stopped after 40 epochs of no improvement, thereby preventing unnecessary computational overhead and potential overfitting.

TerminateOnNaN: This callback, named terminate, ensures that training terminates if any NaN (not a number) values are encountered during training. This is a precautionary measure to prevent invalid calculations or model instability.

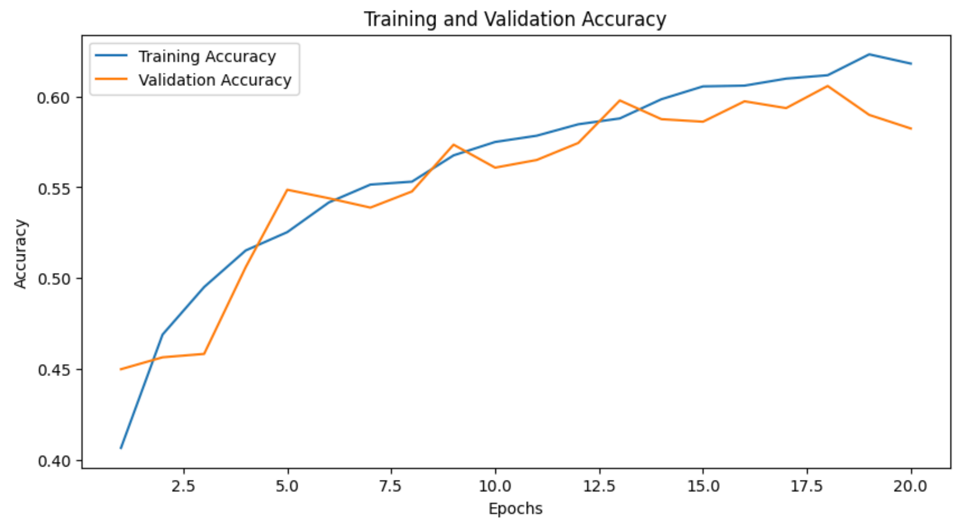
These callbacks are instrumental in enhancing the efficiency and effectiveness of the training process, enabling the model to achieve optimal performance while mitigating the risk of overfitting and instability.

Results - Improved FeedForward Neural Network

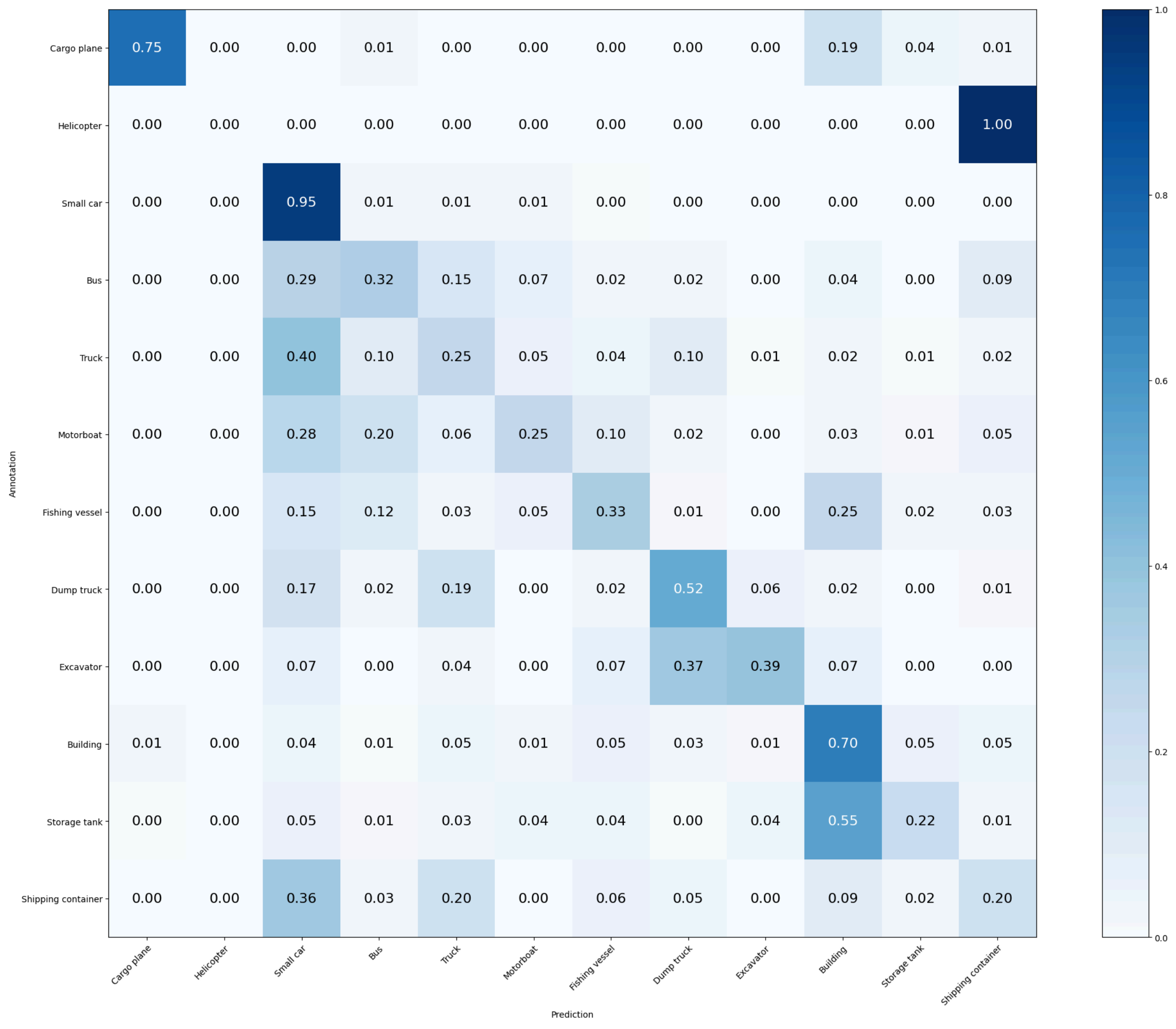
During the training process of the neural network model, the best validation performance was achieved at epoch 18, where the validation accuracy reached **0.6057062745094299**.



The loss curves of the training and validation datasets exhibit a comparable overall trajectory, suggesting similar learning patterns. However, there remains a consistent gap between them, indicating that the model's performance on the validation set may not fully mirror its performance on the training data.



The accuracy curves show the difference between the accuracy of Training and Validation during the epochs. The accuracy is better than the Version 1 but there is still an overfitting problem.



The confusion matrix reveals a noticeable difficulty in correctly classifying instances belonging to the "helicopter" class, with a relatively high rate of misclassification. Additionally, the model exhibits limited proficiency in discerning instances from the "Truck," "Motorboat," and "Shipping container" classes, as evidenced by lower-than-desired recognition rates. Conversely, it demonstrates comparatively better performance in identifying instances from the "Building," "Cargo plane," and "Small car" classes, with relatively higher rates of accurate classification. These observations highlight areas where the model may require further optimization or training to enhance its classification accuracy across all classes.

Mean Accuracy:, it indicates that, on average, the model correctly classified approximately *50.778%* of the instances in the dataset.

Mean Recall: the mean recall of *40.574%* suggests that, on average, the model correctly identified approximately *40.574%* of the true positive instances among all positive instances in the dataset.

Mean Precision: A mean precision of *43.552%* indicates that, on average, approximately *43.552%* of the instances predicted as positive by the model were actually true positive instances.

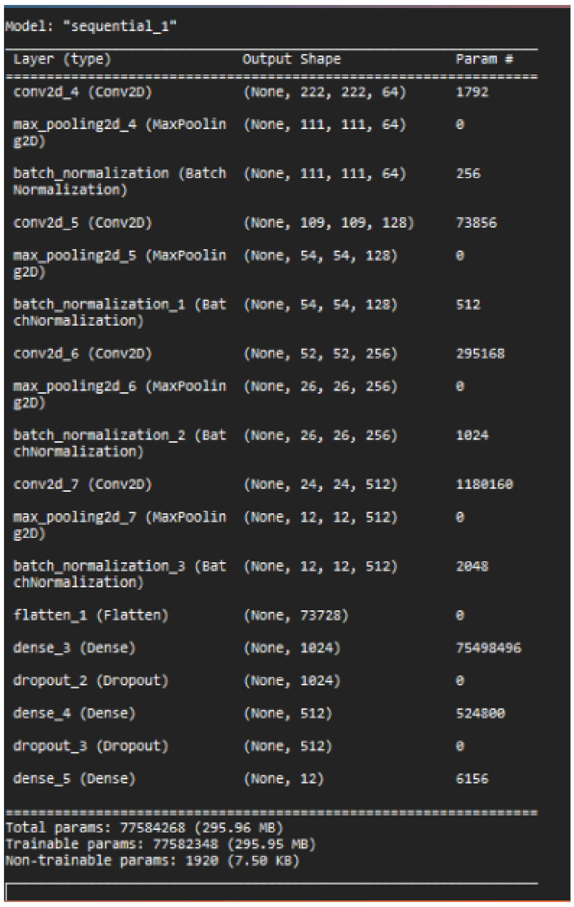
Convolutional Neural Network

As the second step of our project we improve our neural network's performance by adding convolutional layers. Feedforward neural networks (FFNNs) struggle with image data because they don't capture spatial relationships well. Convolutional layers fix this by focusing on local patterns, making our network better at recognizing images accurately, especially with large and complex datasets.

Convolutional Neural Network V1

This network architecture begins with a ***convolutional layer comprising 64 filters***, each with a size of ***3x3***, utilizing ***the rectified linear unit (ReLU)*** activation function. This layer operates on input images sized ***224x224x3.***
Subsequent to each convolutional layer, ***a max-pooling layer*** follows, utilizing a ***2x2 pooling window.*** This operation reduces the spatial dimensions of the feature maps while retaining key information.
Batch normalization layers are incorporated after each max-pooling layer. These layers normalize the input to the activation function of the following layer, enhancing training stability and efficiency.
Upon the final max-pooling layer, the feature maps are flattened into a 1D array to facilitate processing by fully connected layers.

The network then proceeds through a series ***of dense layers.*** The first dense layer consists ***of 1024 neurons with ReLU activation,*** followed by a dropout layer with a dropout rate of 0.5 to mitigate overfitting. The subsequent dense layer comprises ***512 neurons with ReLU*** activation, followed by another dropout layer with the same dropout rate.
The output layer is composed of neurons equal in number to the classification categories. Employing ***the softmax activation*** function, it computes the probability distribution across the categories, indicating the likelihood of each class for a given input image.



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 222, 222, 64)      1792

 max_pooling2d_4 (MaxPoolin  (None, 111, 111, 64)      0
 g2D)

 batch_normalization (Batch  (None, 111, 111, 64)      256
 Normalization)

 conv2d_5 (Conv2D)           (None, 109, 109, 128)     73856

 max_pooling2d_5 (MaxPoolin  (None, 54, 54, 128)       0
 g2D)

 batch_normalization_1 (Bat  (None, 54, 54, 128)       512
 chNormalization)

 conv2d_6 (Conv2D)           (None, 52, 52, 256)       295168

 max_pooling2d_6 (MaxPoolin  (None, 26, 26, 256)       0
 g2D)

 batch_normalization_2 (Bat  (None, 26, 26, 256)       1024
 chNormalization)

 conv2d_7 (Conv2D)           (None, 24, 24, 512)       1180160

 max_pooling2d_7 (MaxPoolin  (None, 12, 12, 512)       0
 g2D)

 batch_normalization_3 (Bat  (None, 12, 12, 512)       2048
 chNormalization)

 flatten_1 (Flatten)         (None, 73728)             0

 dense_3 (Dense)             (None, 1024)              75498496

 dropout_2 (Dropout)         (None, 1024)              0

 dense_4 (Dense)             (None, 512)               524800

 dropout_3 (Dropout)         (None, 512)               0

 dense_5 (Dense)             (None, 12)                6156

=================================================================
Total params: 77584268 (295.96 MB)
Trainable params: 77582348 (295.95 MB)
Non-trainable params: 1920 (7.50 KB)
_________________________________________________________________
```

Model Optimization and Compilation - Convolutional Neural Network V1

The model is compiled using the Adam optimizer with a lower **learning rate of 0.0001**. The categorical cross-entropy loss function is utilized, and accuracy is chosen as the evaluation metric.

Batch Size and Epochs - Convolutional Neural Network V1

The training process utilizes a **batch size of 32**, meaning that each iteration during training processes 32 samples simultaneously. The training loop runs for **30 epochs**, which defines the number of times the entire training dataset is passed forward and backward through the neural network.

Callbacks for Model Training - Convolutional Neural Network V1

ModelCheckpoint tracks the validation accuracy (val_accuracy) and saves the model's weights to a file named 'model.hdf5' whenever there's an improvement, ensuring preservation of the best-performing model.

ReduceLROnPlateau dynamically adjusts the learning rate based on validation accuracy. If no improvement is observed for a specified number of epochs (patience), the learning rate is reduced by a factor of 0.1, facilitating smoother convergence.

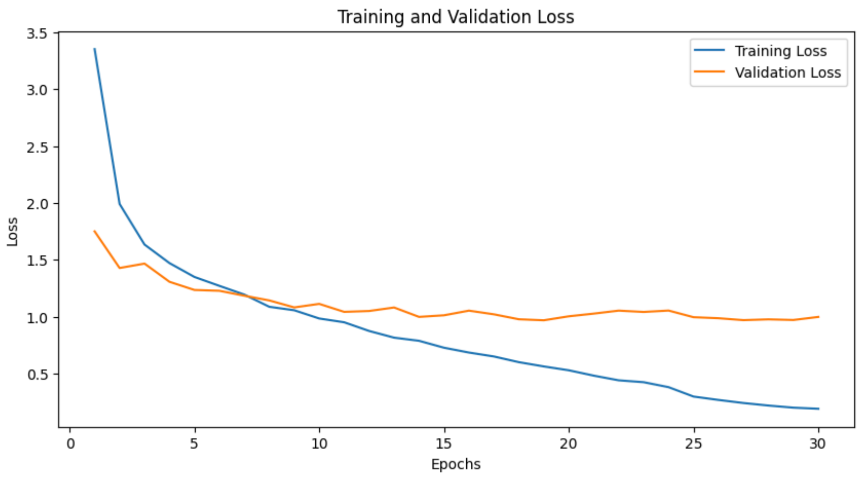
EarlyStopping callback halts training if no improvement in validation accuracy is seen for a defined number of epochs (patience), in this case, 40 epochs. This prevents unnecessary computation and potential overfitting.

TerminateOnNaN callback ensures training halts if any NaN values are encountered, preventing invalid calculations or model instability.

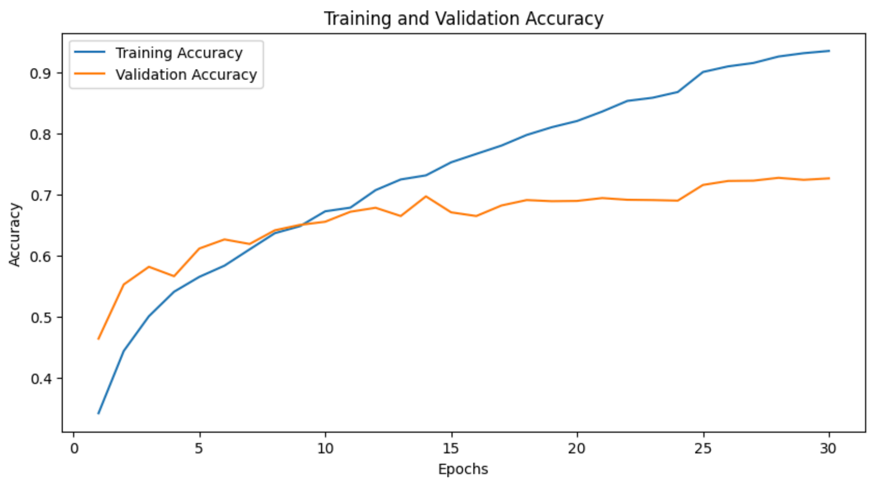
Results - Convolutional Neural Network V1

During the training process of the neural network model, the best validation performance was achieved at epoch 28, where the validation accuracy reached **0.7277829647064209**.

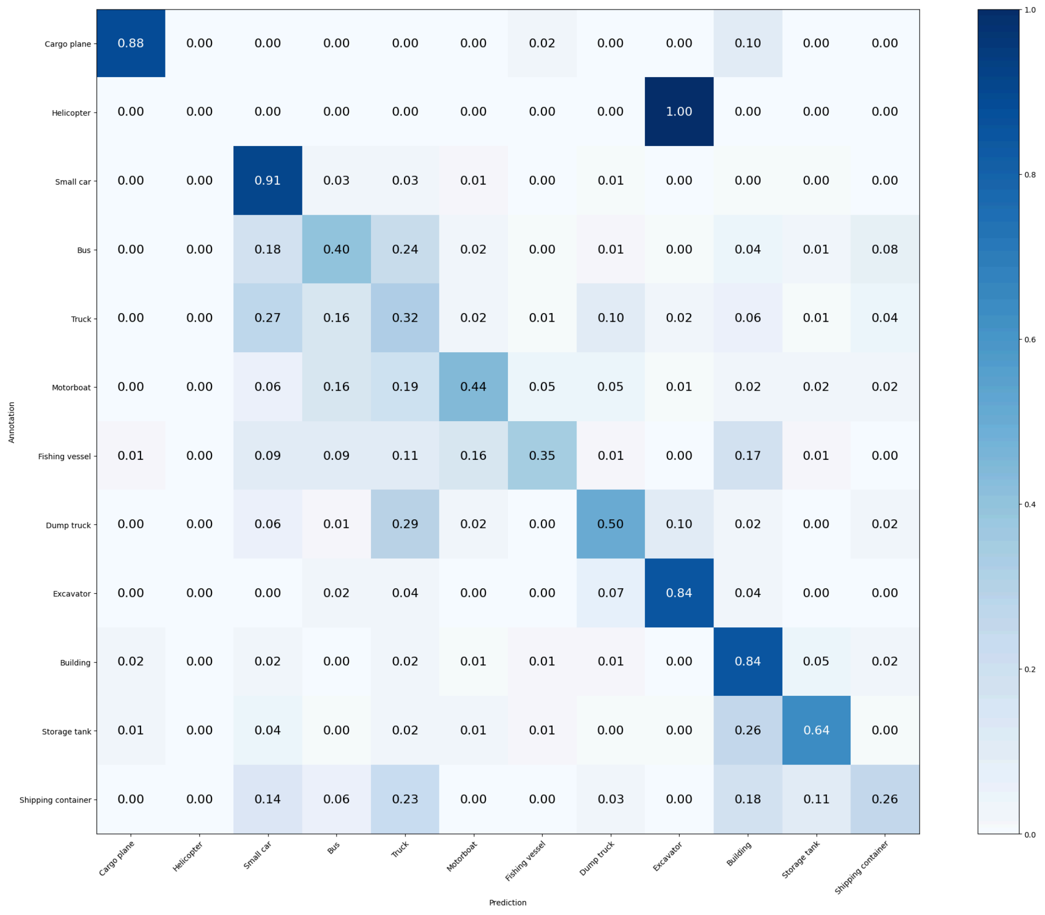
The Training and Validation Loss curves presents a relatively consistent trajectory up to the 15th epoch, after which the gap between them widens significantly in the subsequent epochs.



The training and Validation accuracy curves show an overfitting problem from epoch 15.



The results in the Confusion Matrix are better than the FFNN:



The confusion matrix reveals a noticeable difficulty in correctly classifying instances belonging to the "helicopter" class,the same in the FFNN with a relatively high rate of misclassification. Additionally, the model exhibits limited proficiency in discerning instances from the "Truck," "Fishing Vassel" and "Shipping container" classes, as evidenced by lower-than-desired recognition rates. Conversely, it demonstrates comparatively better performance in identifying instances from the "Building," "Cargo plane" "Excavator", "Small car" classes, with relatively higher rates of accurate classification. These observations highlight areas where the model may require further optimization or training to enhance its classification accuracy across all classes.

Mean Accuracy: With a mean accuracy of 62.846%, it indicates that, on average, the model correctly classified approximately 62.846% of the instances in the dataset.

Mean Recall: The mean recall of 53.256% suggests that, on average, the model correctly identified approximately 53.256% of the true positive instances among all positive instances in the dataset.

Mean Precision: A mean precision of 54.037% indicates that, on average, approximately 54.037% of the instances predicted as positive by the model were actually true positive instances.

Convolutional Neural Network V2

In the context of the second experiment, it's essential to emphasize that the neural network architecture remains entirely unchanged from the preceding experiment. The sole divergence pertains to the implementation of data augmentation techniques. Specifically, alterations are made to the preprocessing pipeline prior to feeding the data into the network for training. These modifications typically involve augmenting the training dataset through transformations such as rotation, scaling, flipping, and translation, among others. The purpose of employing data augmentation is to artificially expand the diversity and size of the training dataset, thereby enhancing the model's robustness to variations and improving its generalization capabilities. By introducing diverse variations of the original images during training, the model is exposed to a broader range of scenarios, facilitating better adaptation to real-world data. Consequently, while the network architecture remains consistent across experiments, the implementation of data augmentation serves as a pivotal factor influencing the model's performance and its ability to effectively generalize to unseen data.

Data Augmentation - Neural Network V2

In the augmentation process applied to the entire dataset, several transformation parameters are utilized to artificially expand the diversity and size of the training dataset. These parameters include:

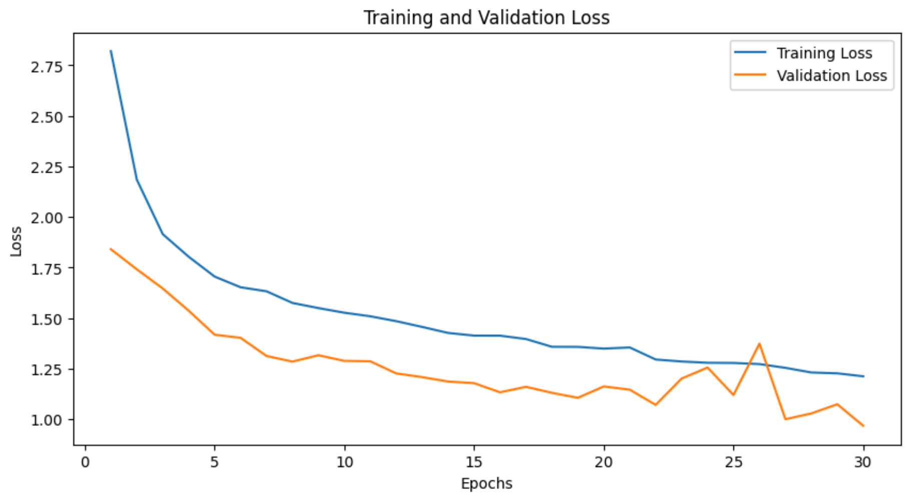
- ***Rotation Range:*** Images are randomly rotated within a range of ±20 degrees, introducing variations in orientation.
- ***Width Shift Range:*** Random horizontal shifts of up to 20% of the image width are applied, simulating changes in object positioning within the image.
- ***Height Shift Range:*** Random vertical shifts of up to 20% of the image height are introduced, mimicking changes in object positioning along the vertical axis.
- ***Horizontal Flip:*** Images are randomly flipped horizontally, creating mirror images and enhancing the model's ability to recognize objects from different perspectives.
- ***Shear Range:*** Random shearing transformations of up to 20% are applied, distorting the shape of objects within the image.

- ***Zoom Range:*** Random zooming of up to 20% is performed, magnifying or reducing the size of objects within the image.
- ***Fill Mode:*** The 'nearest' fill mode is used to handle any empty regions created by the aforementioned transformations, ensuring that these regions are filled appropriately.

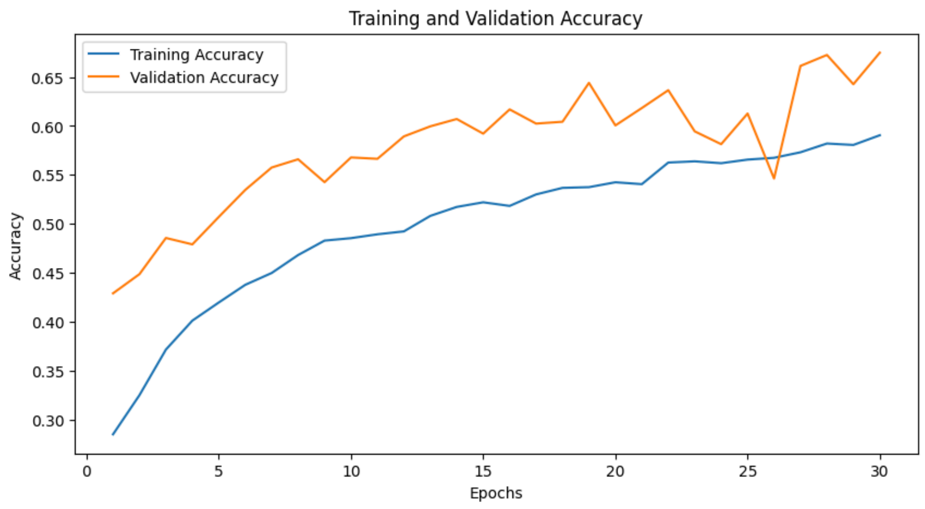
Results - Convolutional Neural Network V2

The best validation model was achieved at epoch 30, with a validation accuracy of **0.6749298572540283%**.

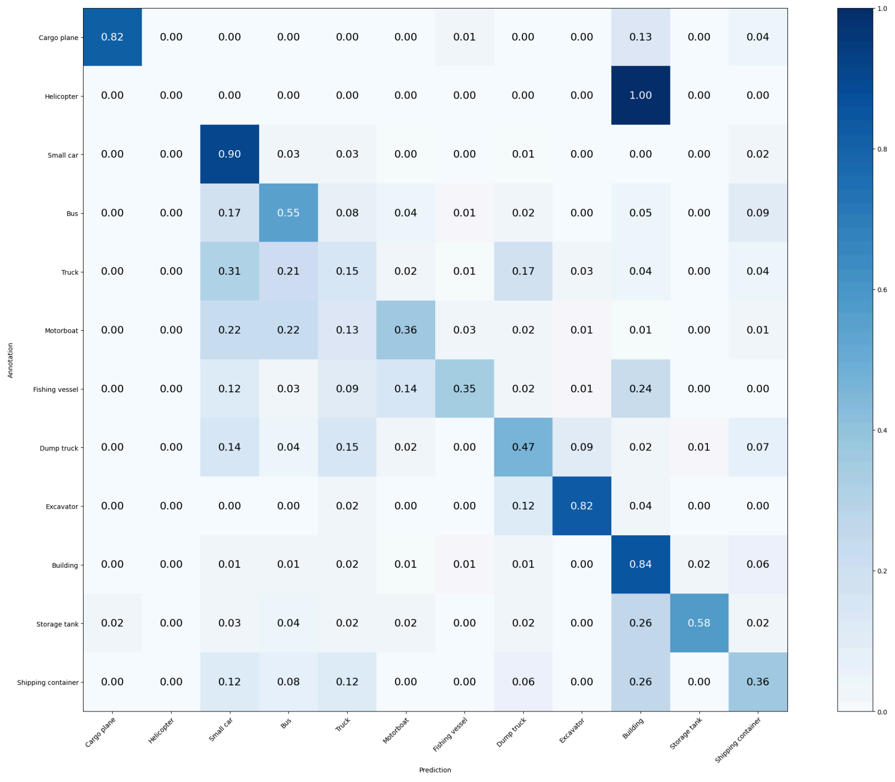
The Training and Validation Loss curves have worse results than the first one without data Agumentation.



The net goes in underfitting as we can see from the Training and Validation Accuracy curves.



In this case, analyzing the Confusion Matrix, the data augmentation didn’t resolve the misclassification of the ‘Helicopter’ class. The Net performs really well in the same classes of



	Cargo plane	Helicopter	Small car	Bus	Truck	Motorboat	Fishing vessel	Dump truck	Excavator	Building	Storage tank	Shipping container
Cargo plane	0.82	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.13	0.00	0.04
Helicopter	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
Small car	0.00	0.00	0.90	0.03	0.03	0.00	0.00	0.01	0.00	0.00	0.00	0.02
Bus	0.00	0.00	0.17	0.55	0.08	0.04	0.01	0.02	0.00	0.05	0.00	0.09
Truck	0.00	0.00	0.31	0.21	0.15	0.02	0.01	0.17	0.03	0.04	0.00	0.04
Motorboat	0.00	0.00	0.22	0.22	0.13	0.36	0.03	0.02	0.01	0.01	0.00	0.01
Fishing vessel	0.00	0.00	0.12	0.03	0.09	0.14	0.35	0.02	0.01	0.24	0.00	0.00
Dump truck	0.00	0.00	0.14	0.04	0.15	0.02	0.00	0.47	0.09	0.02	0.01	0.07
Excavator	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.12	0.82	0.04	0.00	0.00
Building	0.00	0.00	0.01	0.01	0.02	0.01	0.01	0.01	0.00	0.84	0.02	0.06
Storage tank	0.02	0.00	0.03	0.04	0.02	0.02	0.00	0.02	0.00	0.26	0.58	0.02
Shipping container	0.00	0.00	0.12	0.08	0.12	0.00	0.00	0.06	0.00	0.26	0.00	0.36

Mean Accuracy: With a mean accuracy of 60.228%, the model correctly classified approximately 60.228% of the instances in the dataset, on average.

Mean Recall: The mean recall of 51.788% suggests that, on average, the model correctly identified approximately 51.788% of the true positive instances among all positive instances in the dataset.

Mean Precision: A mean precision of 53.939% indicates that, on average, approximately 53.939% of the instances predicted as positive by the model were actually true positive instances.

Convolutional Neural Network V3

In the context of the third experiment, it's essential to emphasize that the underlying neural network architecture remains identical to the preceding experiments. The sole divergence lies in the augmentation methodology applied to the dataset. Specifically, the augmentation in the third experiment encompasses a broader array of transformations and potentially involves a larger number of parameters compared to the second experiment.
Another difference is the number of epochs of the Training, reduced to 20 epochs because of the computational limitation.

Data Augmentation - Neural Network V3

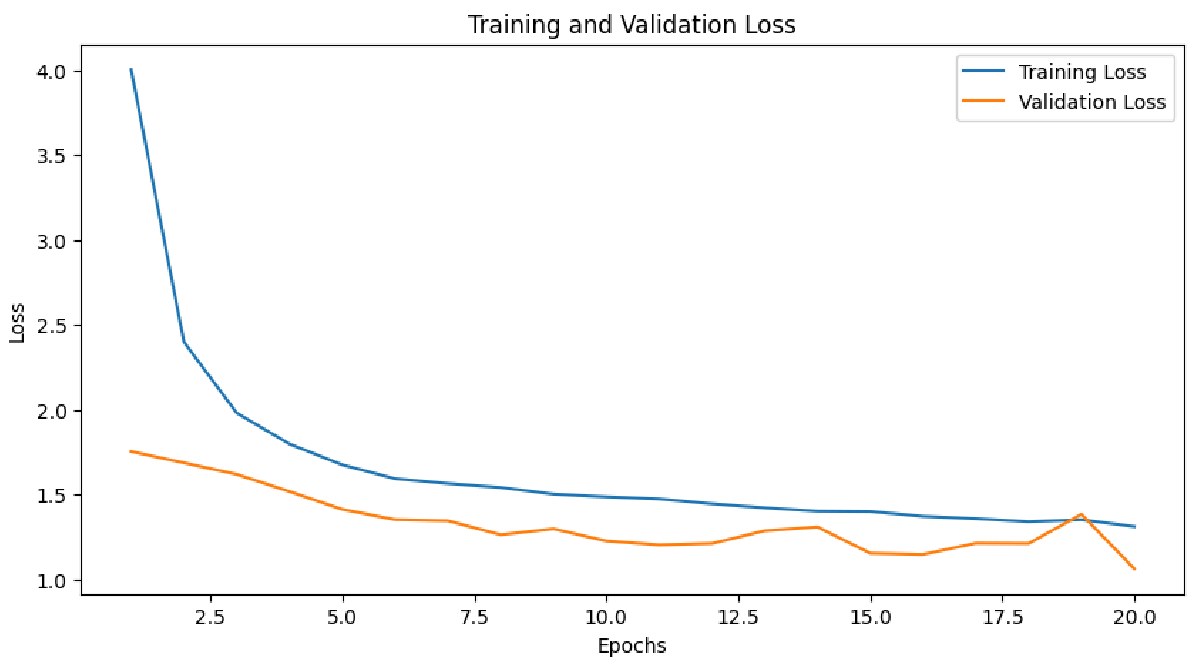
The augmentation process in the third experiment involves a larger number of parameters due to the increased complexity and diversity of the transformations applied.

In the augmentation process applied to the dataset for the third experiment, a more extensive set of transformation parameters is utilized. These parameters include:

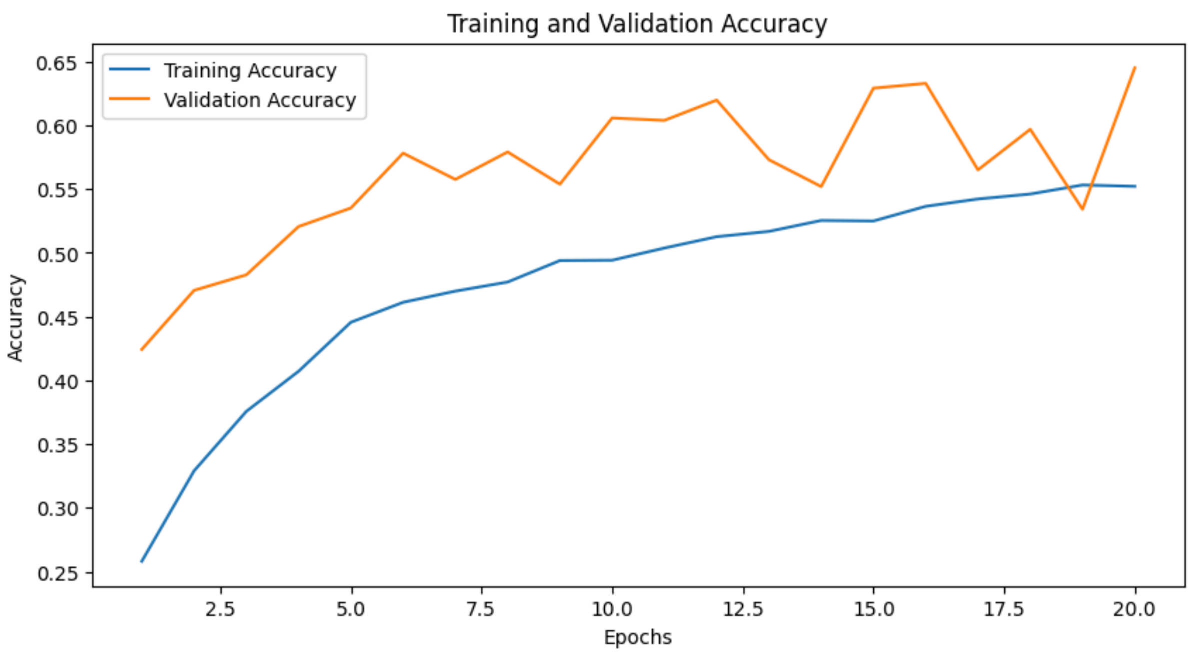
- ***Rotation Range:*** Images are randomly rotated within a range of ±30 degrees, introducing variations in orientation.
- ***Width Shift Range:*** Random horizontal shifts of up to 10% of the image width are applied, simulating changes in object positioning within the image.
- ***Height Shift Range: R***andom vertical shifts of up to 10% of the image height are introduced, mimicking changes in object positioning along the vertical axis.
- ***Horizontal Flip:*** Images are randomly flipped horizontally, creating mirror images and enhancing the model's ability to recognize objects from different perspectives.
- ***Shear Range:*** Random shearing transformations of up to 20% are applied, distorting the shape of objects within the image.
- ***Zoom Range:*** Random zooming of up to 10% is performed, magnifying or reducing the size of objects within the image.
- ***Brightness Range:*** Random adjustments to the brightness of the images within the range of 0.5 to 1.5 are applied, simulating variations in lighting conditions.
- ***Fill Mode:*** The 'nearest' fill mode is used to handle any empty regions created by the aforementioned transformations, ensuring that these regions are filled appropriately.
- ***Cval:*** Value used for filling in newly created pixels if the fill mode is 'constant'.
- ***Vertical Flip:*** Images are randomly flipped vertically, introducing additional variations.
- ***Rescale:*** Images are rescaled by dividing every pixel value by 255, bringing them into the range [0, 1].
- ***Data Format:*** The format of the output data. 'channels_last' means that the color channels are the last dimension of the input data.
- ***Validation Split:*** Fraction of the data to reserve as validation set.
- ***Interpolation Order:*** Order of interpolation to be used for resizing images.

Results - Convolutional Neural Network V3

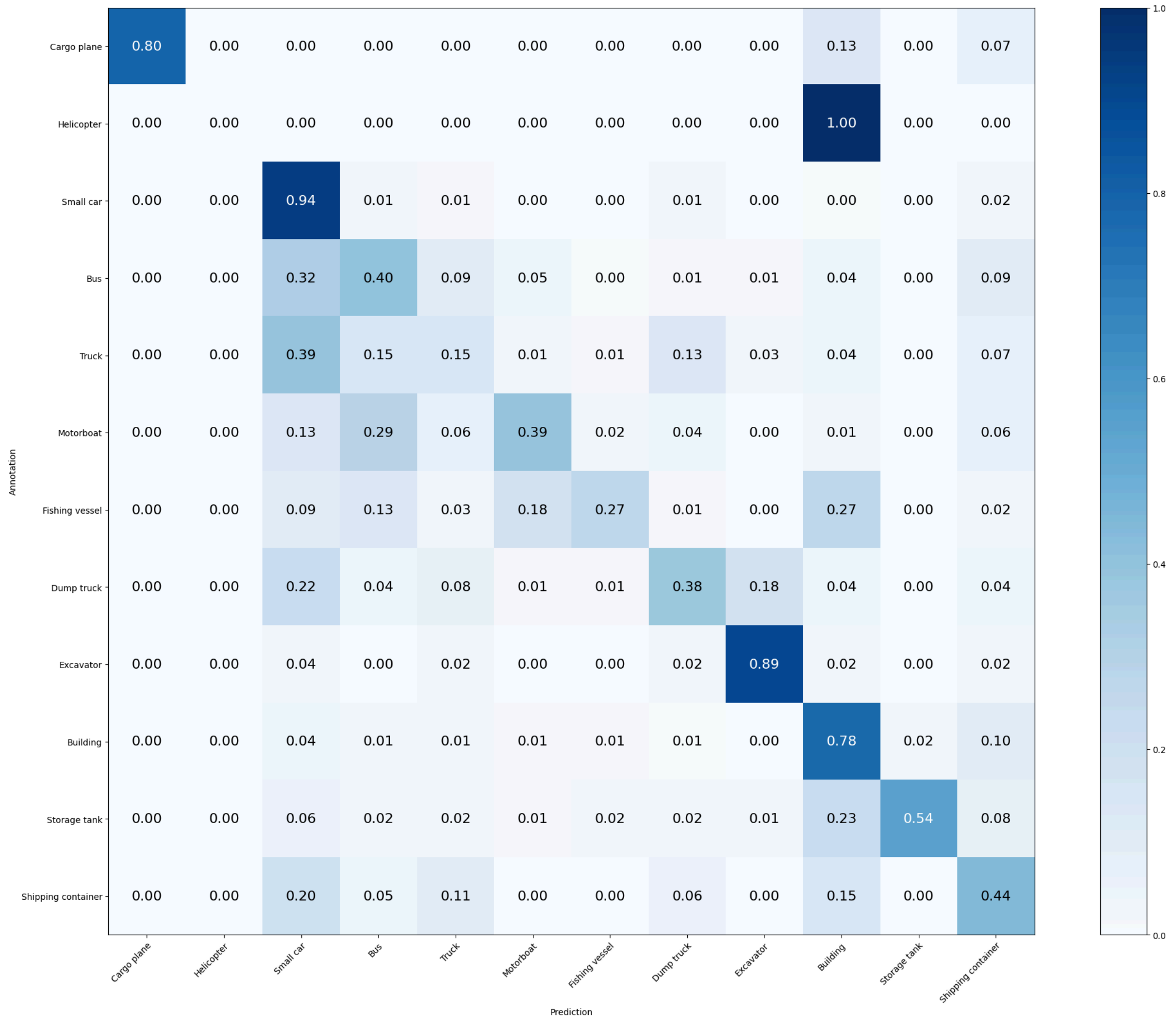
In this case, the model at epoch 30 with a validation accuracy of **0.6749298572540283%**. Loss Curves show more loss in Training than in Validation, it can be a sign of underfitting.



Underfitting confirmed in Accuracy curves.



The confusion Matrix shows a really good accuracy in “Cargo Plane”,”Small Car”, “Excavator” classes. Total Misclasification for the “Helicopter” class, really low accuracy for “Track” and “Fishing Vessel” class.



Annotation \ Prediction	Cargo plane	Helicopter	Small car	Bus	Truck	Motorboat	Fishing vessel	Dump truck	Excavator	Building	Storage tank	Shipping container
Cargo plane	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.07
Helicopter	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
Small car	0.00	0.00	0.94	0.01	0.01	0.00	0.00	0.01	0.00	0.00	0.00	0.02
Bus	0.00	0.00	0.32	0.40	0.09	0.05	0.00	0.01	0.01	0.04	0.00	0.09
Truck	0.00	0.00	0.39	0.15	0.15	0.01	0.01	0.13	0.03	0.04	0.00	0.07
Motorboat	0.00	0.00	0.13	0.29	0.06	0.39	0.02	0.04	0.00	0.01	0.00	0.06
Fishing vessel	0.00	0.00	0.09	0.13	0.03	0.18	0.27	0.01	0.00	0.27	0.00	0.02
Dump truck	0.00	0.00	0.22	0.04	0.08	0.01	0.01	0.38	0.18	0.04	0.00	0.04
Excavator	0.00	0.00	0.04	0.00	0.02	0.00	0.00	0.02	0.89	0.02	0.00	0.02
Building	0.00	0.00	0.04	0.01	0.01	0.01	0.01	0.01	0.00	0.78	0.02	0.10
Storage tank	0.00	0.00	0.06	0.02	0.02	0.01	0.02	0.02	0.01	0.23	0.54	0.08
Shipping container	0.00	0.00	0.20	0.05	0.11	0.00	0.00	0.06	0.00	0.15	0.00	0.44

Mean Accuracy: With a mean accuracy of 57.837%, the model correctly classified approximately 57.837% of the instances in the dataset, on average.

Mean Recall: The mean recall of 49.765% suggests that, on average, the model correctly identified approximately 49.765% of the true positive instances among all positive instances in the dataset.

Mean Precision: A mean precision of 52.735% indicates that, on average, approximately 52.735% of the instances predicted as positive by the model were actually true positive instances.

Transfer Learning

As the third step of the project we improve our neural network's performance using the ***Transfer Learning technique. Transfer learning*** is a technique where a model trained on one task is repurposed for another related task. It involves utilizing a ***pre-trained CNN model*** on a ***large dataset***, typically for a generic task like image classification, and then adapting this ***pre-trained model*** to a different but related task.

Pre-Trained CCN

VGG16

VGG16, which stands for Visual Geometry Group (VGG), stands as a cornerstone in the realm of computer vision, renowned for its depth and precision. With its 16 convolutional layers, VGG16 has redefined the landscape of object recognition, establishing new standards for accuracy and performance.
Originally conceived as a deep neural network, VGG16 has solidified its position as one of the most influential architectures in image recognition. Its architecture emphasizes depth, featuring a multitude of stacked layers that enable the model to extract intricate patterns and features from raw image data with unparalleled efficiency.
In our application, we harnessed the power of VGG16 for its exceptional prowess in image recognition tasks. By integrating this esteemed architecture into our framework, we capitalized on its proven ability to handle diverse datasets and deliver robust performance.
Our decision to adopt VGG16 was further reinforced by its remarkable performance metrics. Notably, VGG16 achieves an impressive top-5 test accuracy of nearly 92.7% on ImageNet, a comprehensive dataset comprising millions of images spanning various categories.

EfficientNetB7

EfficientNet, a breakthrough convolutional neural network (CNN) architecture introduced by Google in 2019, stands out as a pioneering solution meticulously engineered to achieve unparalleled accuracy while optimizing computational efficiency.

Central to EfficientNet's innovation is its unique approach to scaling the network architecture. Unlike traditional methods that focus on individual dimensions like depth or width, EfficientNet harmoniously scales across depth, width, and resolution simultaneously. This holistic scaling strategy ensures an optimal balance between model complexity and computational resources, making it an ideal choice for various applications.

In our specific implementation, we made a deliberate choice to leverage EfficientNetB7, the largest variant within the EfficientNet family. EfficientNetB7 offers a remarkable balance between computational efficiency and performance, making it well-suited for our task. By selecting EfficientNetB7, we aimed to harness its formidable capabilities to achieve state-of-the-art results while maintaining computational feasibility.

VGG19

VGG19 stands as a prominent convolutional neural network architecture crafted by the Visual Geometry Group (VGG) at the University of Oxford. Serving as an extension to its predecessor, VGG16, this model boasts 19 layers, hence its moniker.
The architecture of VGG19 is characterized by a stack of 16 convolutional layers, accompanied by 3 fully connected layers. These convolutional layers are organized into five distinct blocks, each comprising multiple convolutional layers, interspersed with max-pooling layers that facilitate down-sampling.
In its convolutional layers, VGG19 employs compact 3x3 filters with a stride of 1 pixel, coupled with rectified linear activation functions (ReLU). The number of filters escalates with the network's depth, commencing with 64 filters in the inaugural layer and doubling post each max-pooling layer.
Pooling layers play a pivotal role in VGG19, leveraging 2x2 filters and a stride of 2 pixels to condense the spatial dimensions of feature maps while retaining salient information.
The culmination of VGG19 lies in its fully connected layers, the final three in particular. These layers harbor 4096 neurons each, culminating in an output layer responsible for generating predictions. Dropout layers are judiciously inserted post these fully connected layers to mitigate overfitting.
Throughout the network, the activation function of choice is ReLU (Rectified Linear Unit), with softmax activation commonly adopted at the output layer for multi-class classification endeavors.
VGG19 enjoys widespread adoption as a pre-trained model for diverse computer vision tasks. Pre-trained weights, honed on extensive image datasets such as ImageNet, are readily accessible, facilitating transfer learning initiatives aimed at novel tasks, even in scenarios where data is limited.

EfficientNetV2L

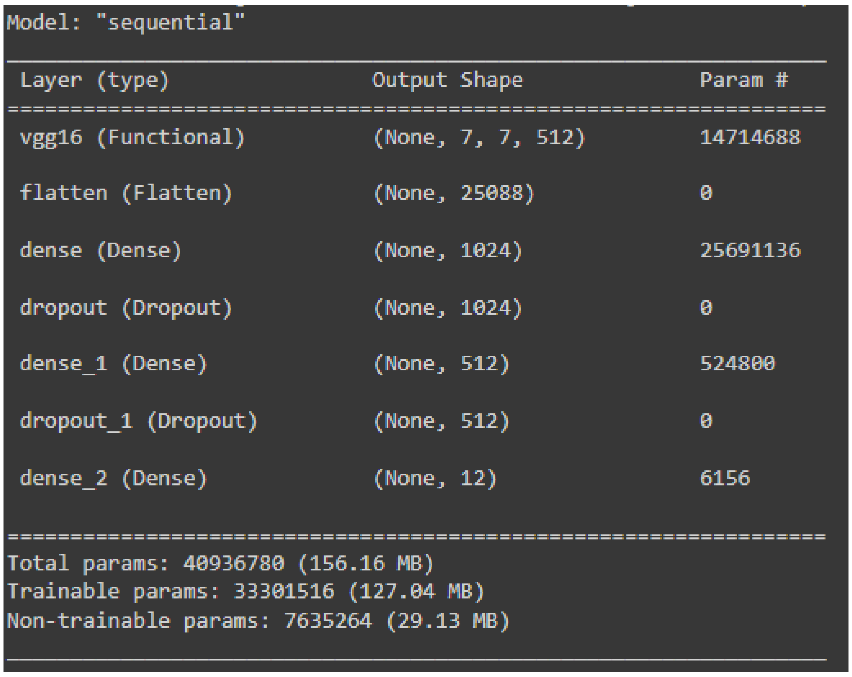
EfficientNetV2L builds upon the foundation laid by the EfficientNet family of neural networks, known for their remarkable balance between computational efficiency and high performance across various computer vision tasks. At its core, EfficientNetV2L retains the efficient architectural principles that made its predecessors successful.
The backbone of EfficientNetV2L is rooted in the EfficientNet architecture, which is characterized by a deep and narrow network structure. This backbone has been meticulously crafted to strike a delicate balance between model performance and computational efficiency. It achieves this balance through a process known as scaling, which involves systematically adjusting the network's depth, width, and resolution to achieve optimal performance under resource constraints.
EfficientNetV2L extends upon this foundation by incorporating additional architectural enhancements aimed at further improving its efficiency and performance. These enhancements may include novel design choices in convolutional layers, attention mechanisms, or regularization techniques. By leveraging these advancements, EfficientNetV2L is able to achieve state-of-the-art performance on a wide range of computer vision tasks while remaining highly resource-efficient.

Large Dataset

The large dataset utilized for transfer learning in our application is ***ImageNet.*** ImageNet is a comprehensive dataset comprising millions of labeled images across thousands of categories. It serves as a benchmark dataset in the field of computer vision, facilitating the training and evaluation of image recognition models. ImageNet spans a diverse range of objects, scenes, and concepts, providing a rich source of visual data for machine learning tasks. Each image in ImageNet is annotated with class labels, allowing models to learn to distinguish between different objects and categories. With its extensive coverage and meticulous labeling, ImageNet enables researchers and practitioners to develop and benchmark state-of-the-art algorithms for various computer vision tasks, including image classification, object detection, and scene understanding.

Transfer Learning Network V1

The model is set up for image classification using transfer learning with the VGG16 model. It imports necessary libraries from TensorFlow and its Keras API. The VGG16 model is loaded with pre-trained ImageNet weights, excluding its fully connected layers. The layers of the ***pre-trained VGG16*** model up to the last four layers are frozen to retain pre-trained knowledge. A new model is created using the Sequential API from Keras, with the pre-trained VGG16 model as the base. Additional layers are added on top, including ***Flatten, Dense with ReLU activation*** and ***L2 regularization***, and ***Dropout(0.5) layers.*** The final dense layer employs ***a softmax activation function*** for multi-class classification problems.


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 1024)              25691136

 dropout (Dropout)           (None, 1024)              0

 dense_1 (Dense)             (None, 512)               524800

 dropout_1 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 12)                6156

=================================================================
Total params: 40936780 (156.16 MB)
Trainable params: 33301516 (127.04 MB)
Non-trainable params: 7635264 (29.13 MB)
_________________________________________________________________
```

Model Optimization and Compilation - Transfer Learning Network V1

The model is optimized and compiled using the Adam optimizer with ***learning rate*** set to ***0.0001***. This optimizer is chosen due to its effectiveness in handling sparse gradients and its adaptability to different types of data. The ***categorical cross-entropy loss function*** is employed, which is well-suited for multi-class classification tasks. Accuracy is selected as the evaluation metric to assess the model's performance during training and validation.

Batch Size and Epochs - Transfer Learning Network V1

During training, a ***batch size of 32*** is utilized, meaning that 32 samples are processed simultaneously in each iteration of the training loop. This batch size strikes a balance between computational efficiency and model convergence. The training process is executed over ***30 epochs***, determining the number of complete passes through the entire training dataset. This choice of epochs allows the model to sufficiently learn the underlying patterns and features within the data.

Callbacks for Model Training -Transfer Learning Network V1

Several callbacks are incorporated to enhance the efficiency and stability of the training process:

ModelCheckpoint: This callback monitors the validation accuracy (val_accuracy) and saves the model's weights to a file named 'model.hdf5' whenever there is an improvement in performance. This ensures that the best-performing model configuration is preserved for later use.

ReduceLROnPlateau: This callback dynamically adjusts the learning rate based on changes in validation accuracy. If no improvement is observed for a specified number of epochs (patience), indicated as 10 epochs in this case, the learning rate is reduced by a factor of 0.1. This adaptive learning rate schedule facilitates smoother convergence and prevents the model from getting stuck in local minima.

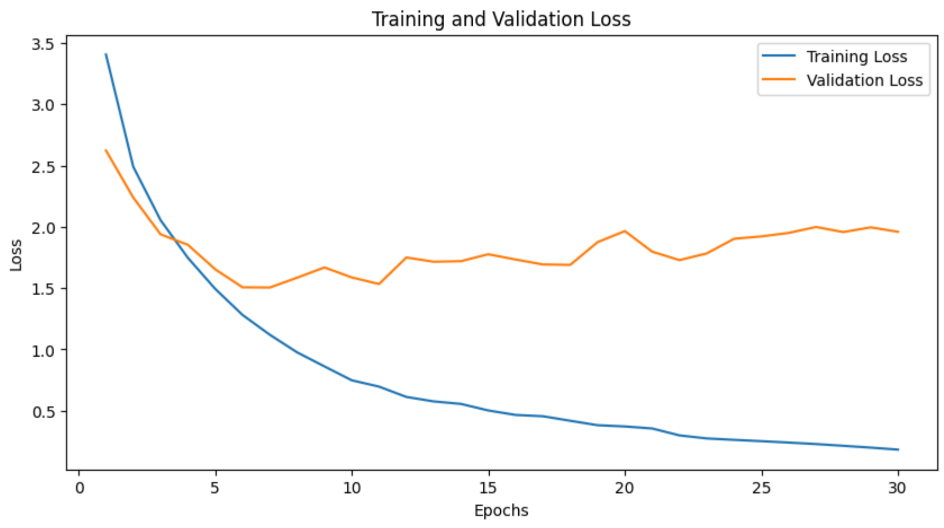
EarlyStopping: This callback terminates the training process if no improvement in validation accuracy is detected for a defined number of epochs, set to 40 epochs in this instance. Early stopping prevents overfitting by halting training when further iterations are unlikely to yield significant improvements in generalization performance.

TerminateOnNaN: This callback ensures training halts immediately if any NaN (Not a Number) values are encountered during computation. This precautionary measure helps maintain the integrity of the training process and prevents invalid calculations or model instability.

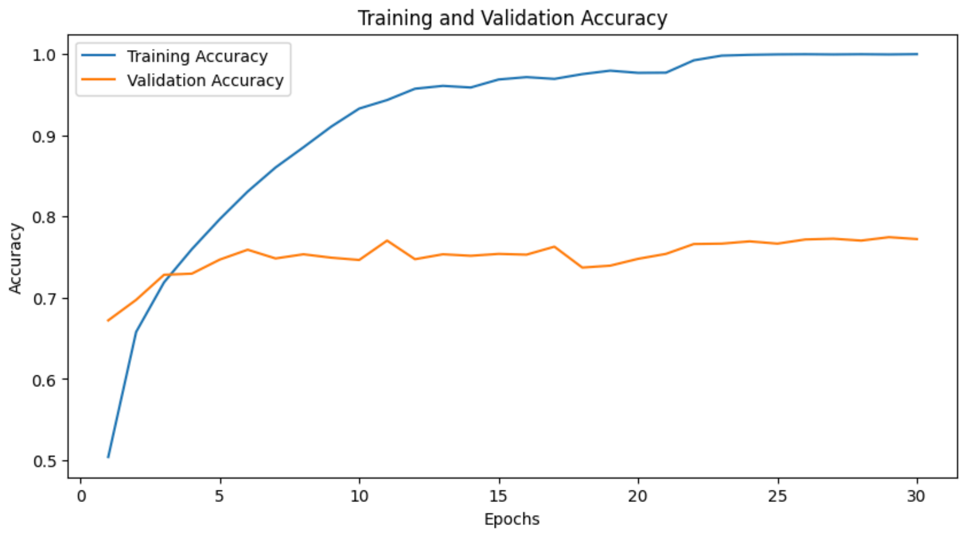
Results - Transfer Learning Network V1

The best validation model was achieved at ***epoch 29***, with a validation accuracy of ***0.7745556831359863%.***

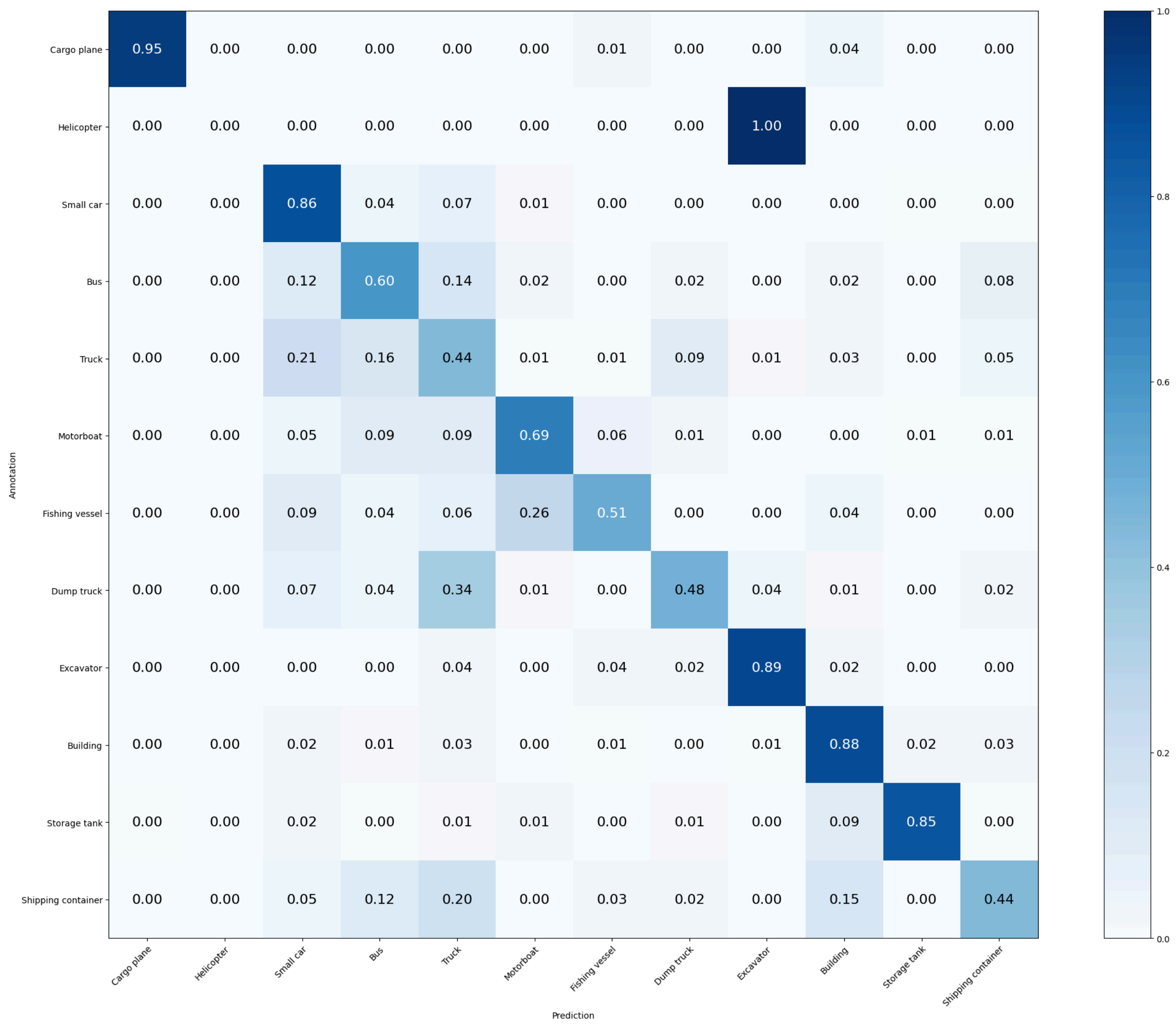
The Loss Curves show a huge gap since the 8 epoch. It means that the model is not well balanced
A



.
In training and validation Accuracy graphics we can see a huge gap between the curves. It means that the Net goes in overfitting.



Analyzing the ***Confusion Matrix,*** the results are better than CNN. The confusion matrix reveals a noticeable difficulty in correctly classifying instances belonging to the "helicopter" class,the same in the FFNN and CNN with a relatively high rate of misclassification. Additionally, the model exhibits limited proficiency in discerning instances from the "Truck," "Dump Truck” and "Shipping container" classes, as evidenced by lower-than-desired recognition rates. Conversely, it demonstrates comparatively better performance in identifying instances from the "Building," "Cargo plane" “Excavator”, "Small car",”Storage Tank” classes, with relatively higher rates of accurate classification.



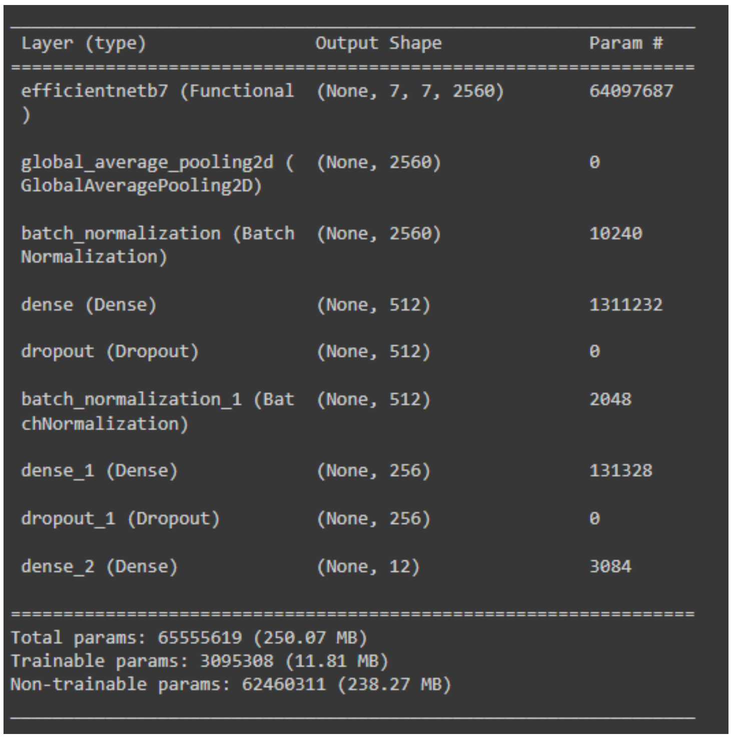
Mean Accuracy: With a mean accuracy of 72.865%, it indicates that, on average, the model correctly classified approximately 72.865% of the instances in the dataset.

Mean Recall: The mean recall of 63.274% suggests that, on average, the model correctly identified approximately 63.274% of the true positive instances among all positive instances in the dataset.

Mean Precision: A mean precision of 63.792% indicates that, on average, approximately 63.792% of the instances predicted as positive by the model were actually true positive instances.

Transfer Learning Network V2

The ***EfficientNetB7*** model is initialized with pre-trained weights obtained from ImageNet, with the exclusion of the fully connected top layers. Subsequently, a new sequential model is constructed, comprising the **EfficientNetB7** base model followed by a global average pooling layer, which diminishes the spatial dimensions of the representation through averaging, and a batch normalization layer, aimed at normalizing the output of the preceding layer. Two dense layers are introduced, each succeeded by a dropout layer to mitigate overfitting. Both **dropout layers** are set with a rate of **0.6.** Moreover, each dense layer employs **L2 regularization** with a coefficient of **0.001** to tackle overfitting issues. Lastly, an output dense layer is appended with a number of units corresponding to the desired number of classification categories, utilizing the softmax activation function to derive probabilities for each class.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetb7 (Functional  (None, 7, 7, 2560)        64097687
 )

 global_average_pooling2d (  (None, 2560)              0
 GlobalAveragePooling2D)

 batch_normalization (Batch  (None, 2560)              10240
 Normalization)

 dense (Dense)               (None, 512)               1311232

 dropout (Dropout)           (None, 512)               0

 batch_normalization_1 (Bat  (None, 512)               2048
 chNormalization)

 dense_1 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 12)                3084

=================================================================
Total params: 65555619 (250.07 MB)
Trainable params: 3095308 (11.81 MB)
Non-trainable params: 62460311 (238.27 MB)
_________________________________________________________________
```

Model Optimization and Compilation - Transfer Learning V2

The model is optimized and compiled using the Adam optimizer with a lower learning rate set to 0.0001. This specific learning rate is chosen to facilitate fine-tuning on the pre-trained EfficientNetB7 model while preventing drastic changes to the already learned features. Adam optimizer is preferred for its robustness in handling sparse gradients and adaptability across various data types. The categorical cross-entropy loss function is employed, given its suitability for multi-class classification tasks. Accuracy is designated as the evaluation metric to gauge the model's performance during both training and validation phases.

Batch Size and Epochs - Transfer Learning Network V2

During training, a batch size of **32** is utilized, meaning that 32 samples are processed simultaneously in each iteration of the training loop. This batch size strikes a balance between computational efficiency and model convergence. The training process is executed over **10 epochs**, determining the number of complete passes through the entire training dataset. This choice of epochs allows the model to sufficiently learn the underlying patterns and features within the data. The minor number of epochs is caused by the limitation for the **T4 GPU** usage.

Callbacks for Model Training - Transfer Learning Network V2

Several callbacks are integrated to improve the efficiency and stability of the training process:

ModelCheckpoint: This callback monitors validation accuracy (val_accuracy) and saves the model weights to 'model.hdf5' file whenever there's an improvement in performance, ensuring preservation of the best-performing model configuration.

ReduceLROnPlateau: This callback dynamically adjusts the learning rate based on changes in validation loss. If no improvement is observed for a specified number of epochs (patience), set to 3 epochs in this case, the learning rate is reduced by a factor of 0.2. This adaptive learning rate schedule aids in smoother convergence and prevents the model from getting stuck in local minima.

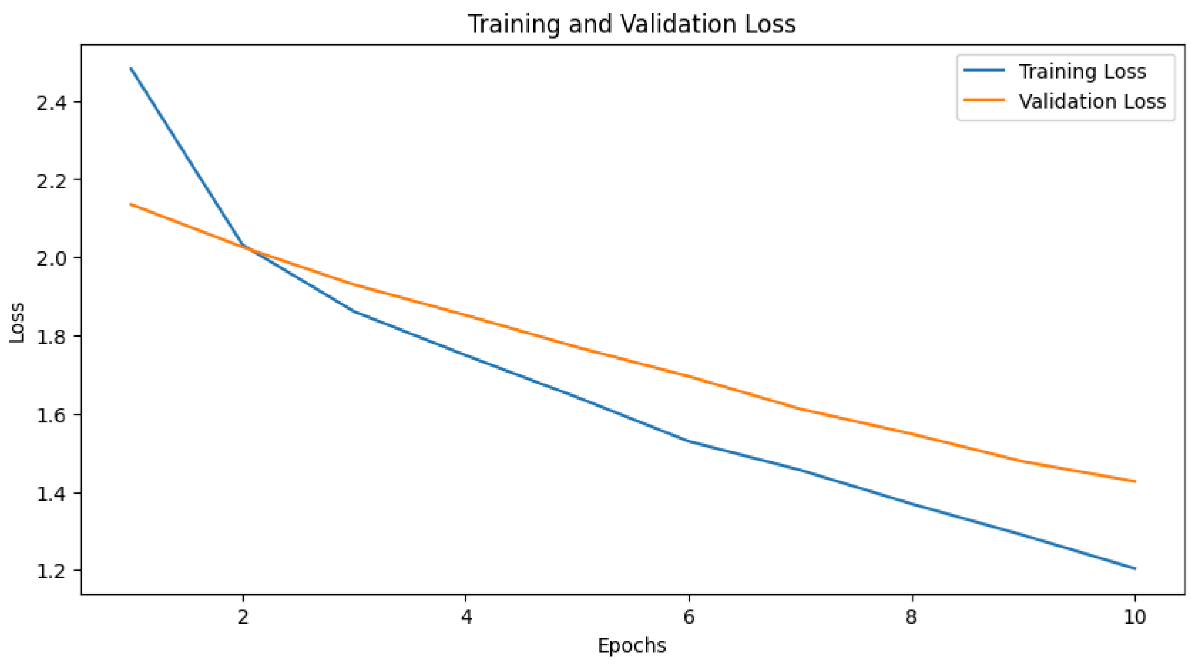
EarlyStopping: This callback terminates the training process if no improvement in validation accuracy is detected for a defined number of epochs, set to 40 epochs. Early stopping prevents overfitting by halting training when further iterations are unlikely to yield significant improvements in generalization performance.

TerminateOnNaN: This callback ensures training halts immediately if any NaN (Not a Number) values are encountered during computation, maintaining the integrity of the training process and preventing invalid calculations or model instability.

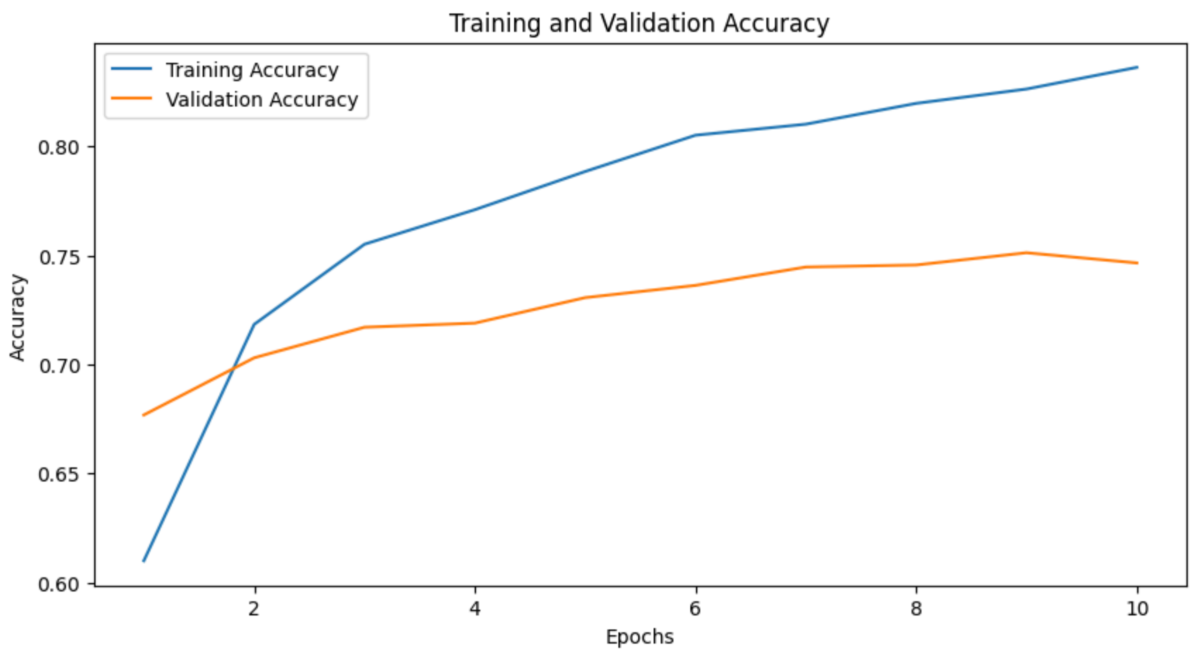
Results - Transfer Learning Network V2

The best validation model was achieved at ***epoch 9***, with a validation accuracy of ***0.7511693239212036%.***

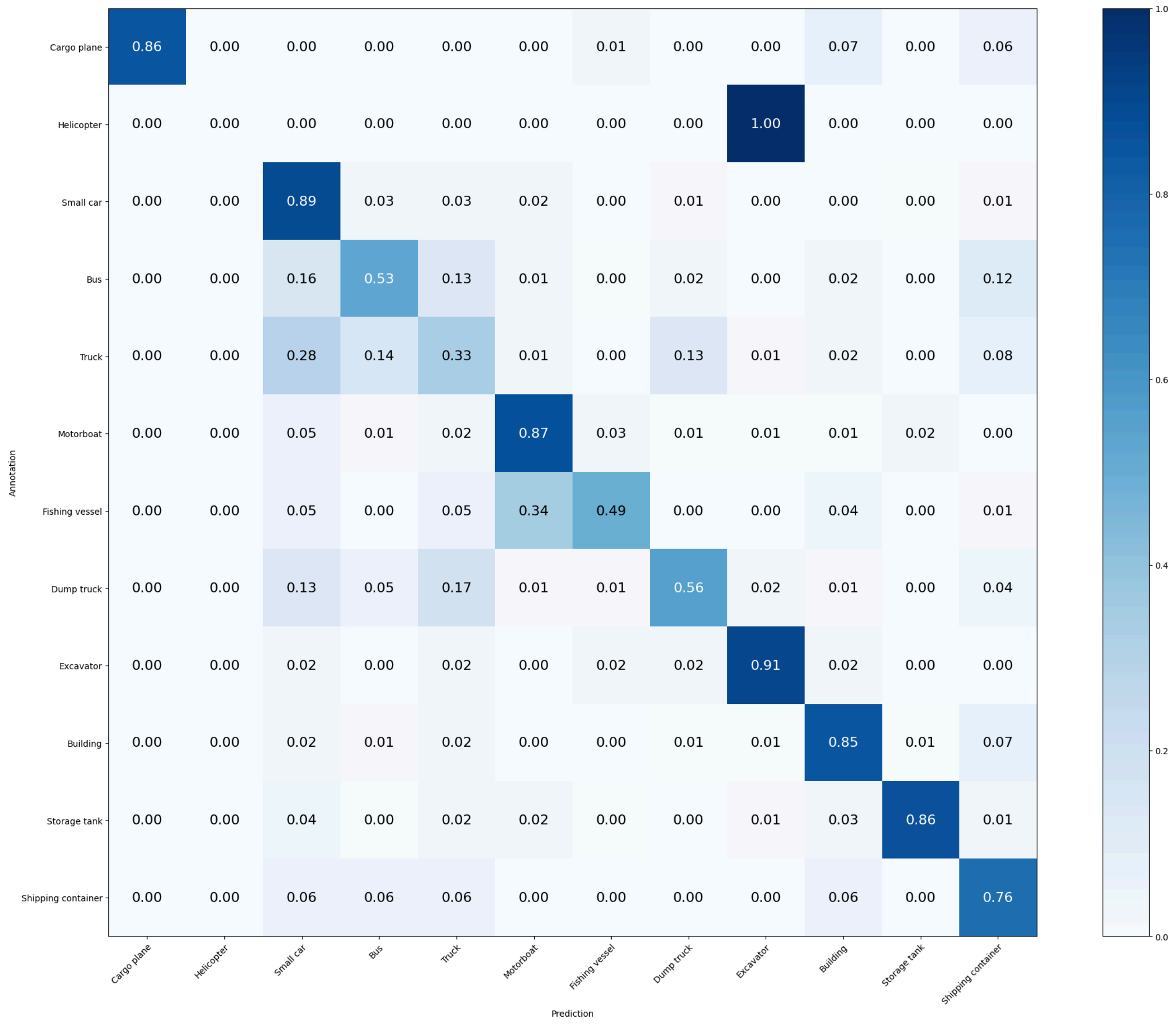
The loss graphs exhibit a similar trend between Training and Validation, indicating a balanced convergence of the model's performance.



The accuracy gap is reduced but the overfitting is still presented.



Analyzing Confusion Matrix, the problem about the ‘Helicopter’ class is still not resolved but the accuracy for almost each class is increased.



Mean Accuracy: With a mean accuracy of 74.459%, it indicates that, on average, the model correctly classified approximately 74.459% of the instances in the dataset.

Mean Recall: The mean recall of 65.883% suggests that, on average, the model correctly identified approximately 65.883% of the true positive instances among all positive instances in the dataset.

Mean Precision: A mean precision of 65.851% indicates that, on average, approximately 65.851% of the instances predicted as positive by the model were actually true positive instances.

Transfer Learning Network V3

The third experiment is a fusion of the methodologies employed in the preceding two experiments. It amalgamates the VGG16 architecture used for Transfer Learning utilized in the first experiment with the data augmentation technique applied exclusively to a single class, namely 'Helicopter,' characterized by the lowest number of samples. The rationale behind this hybrid approach is to enhance the model's performance by leveraging the robustness of the VGG16 network while addressing the imbalance in the dataset through targeted augmentation.
By integrating the VGG16 architecture, which has demonstrated proficiency in image recognition tasks, we aim to capitalize on its established capabilities in extracting meaningful features from visual data. Concurrently, the selective application of data augmentation techniques to the 'Helicopter' class seeks to rectify the imbalance inherent in the dataset.
The other difference between the first one and the third one is the number of epochs: in the first one are 30, in this one are 15 for the computational limitation.

Learning Rate

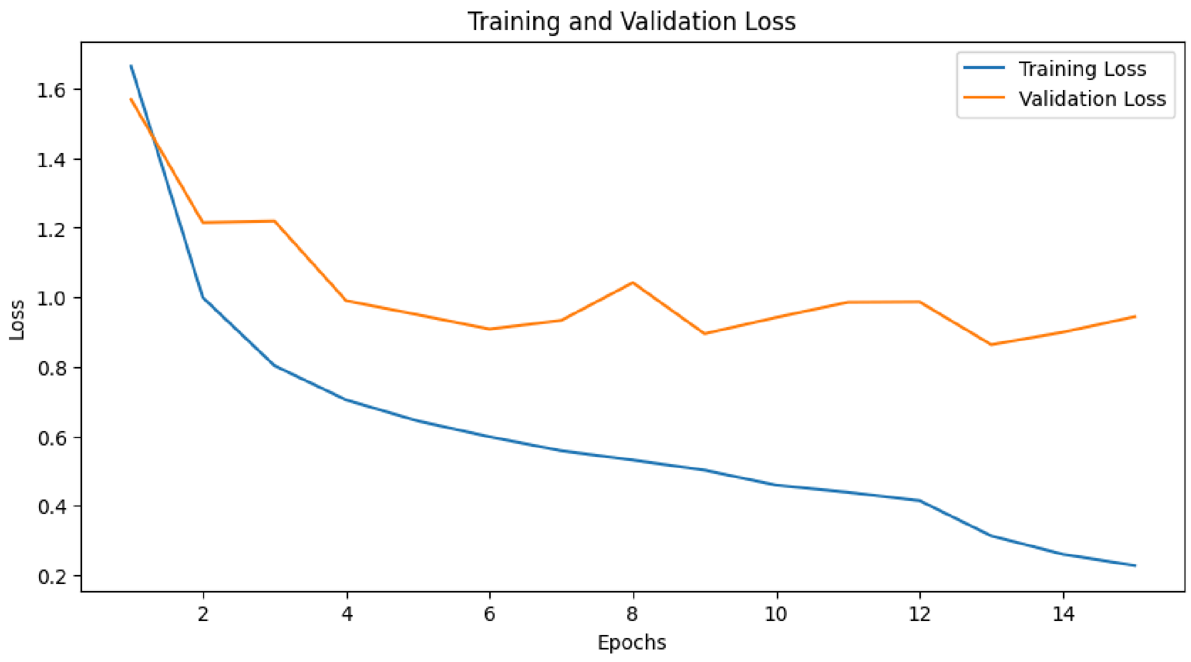
The other difference with the other experiments are the values of the learning rate :

- ***Learning rate:*** 1e-3
- ***Beta 1:*** 0.9
- ***Beta 2:*** 0.999
- ***Epsilon***: 1e-8
- ***AMSGrad***: True
- ***Clipnorm:*** 1.0
- ***Clipvalue:*** 0.5

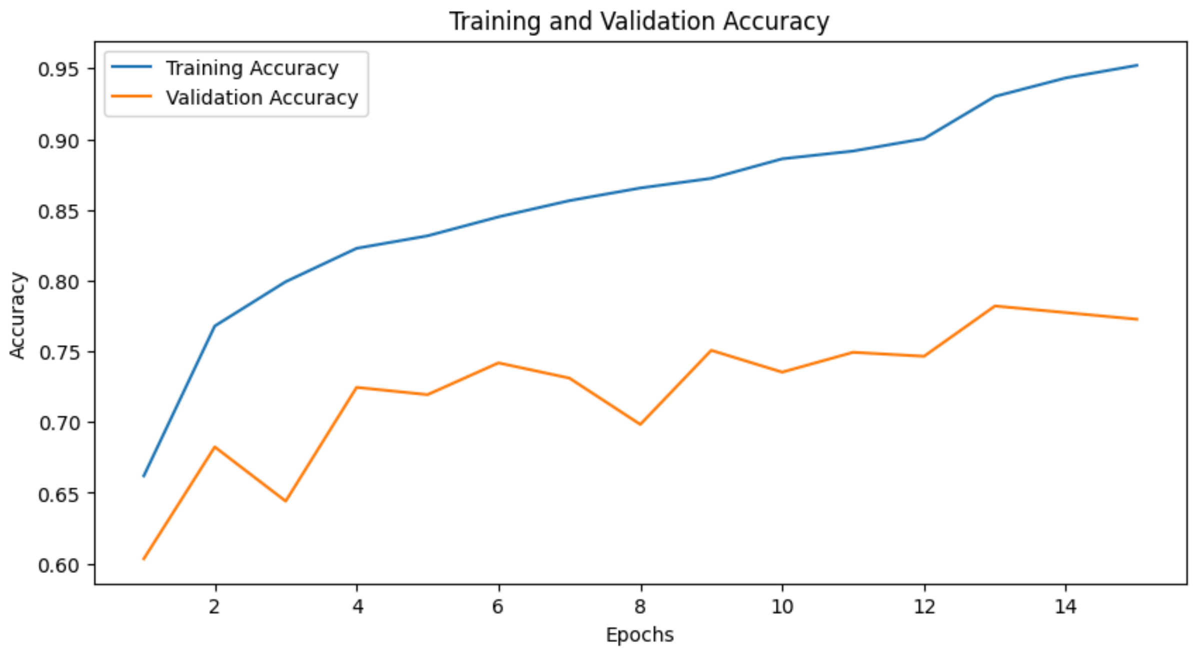
Results - Transfer Learning Network V3

The best validation model was achieved at epoch 13, with a validation accuracy of ***0.7820392847061157***

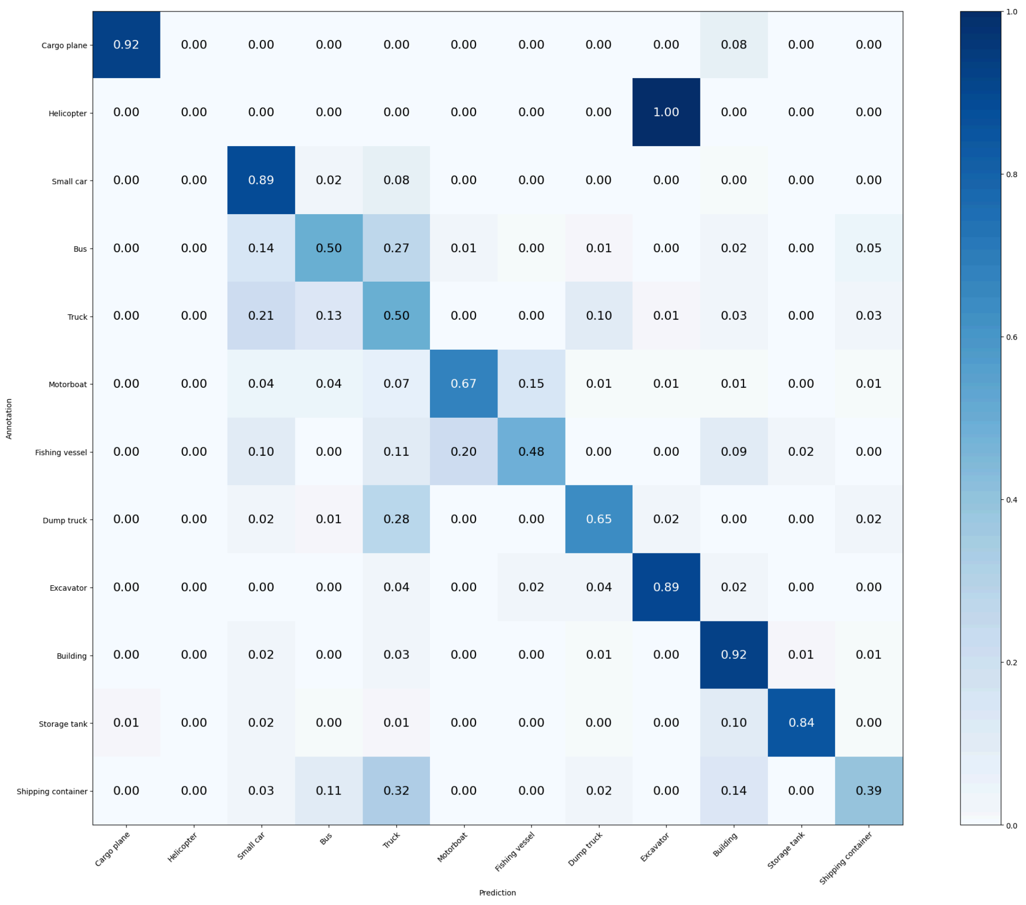
The trend of the loss curves looks better of the first experiment but they are worse than the second one.



Accuracy curves show the still remaining overfitting problem.



Finally the Confusion Matrix shows the total misclassification of the enhanced class ‘Helicopter’ and almost the same accuracy for the other class as the previous experiments.



	Cargo plane	Helicopter	Small car	Bus	Truck	Motorboat	Fishing vessel	Dump truck	Excavator	Building	Storage tank	Shipping container
Cargo plane	0.92	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00	0.00
Helicopter	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
Small car	0.00	0.00	0.89	0.02	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Bus	0.00	0.00	0.14	0.50	0.27	0.01	0.00	0.01	0.00	0.02	0.00	0.05
Truck	0.00	0.00	0.21	0.13	0.50	0.00	0.00	0.10	0.01	0.03	0.00	0.03
Motorboat	0.00	0.00	0.04	0.04	0.07	0.67	0.15	0.01	0.01	0.01	0.00	0.01
Fishing vessel	0.00	0.00	0.10	0.00	0.11	0.20	0.48	0.00	0.00	0.09	0.02	0.00
Dump truck	0.00	0.00	0.02	0.01	0.28	0.00	0.00	0.65	0.02	0.00	0.00	0.02
Excavator	0.00	0.00	0.00	0.00	0.04	0.00	0.02	0.04	0.89	0.02	0.00	0.00
Building	0.00	0.00	0.02	0.00	0.03	0.00	0.00	0.01	0.00	0.92	0.01	0.01
Storage tank	0.01	0.00	0.02	0.00	0.01	0.00	0.00	0.00	0.00	0.10	0.84	0.00
Shipping container	0.00	0.00	0.03	0.11	0.32	0.00	0.00	0.02	0.00	0.14	0.00	0.39

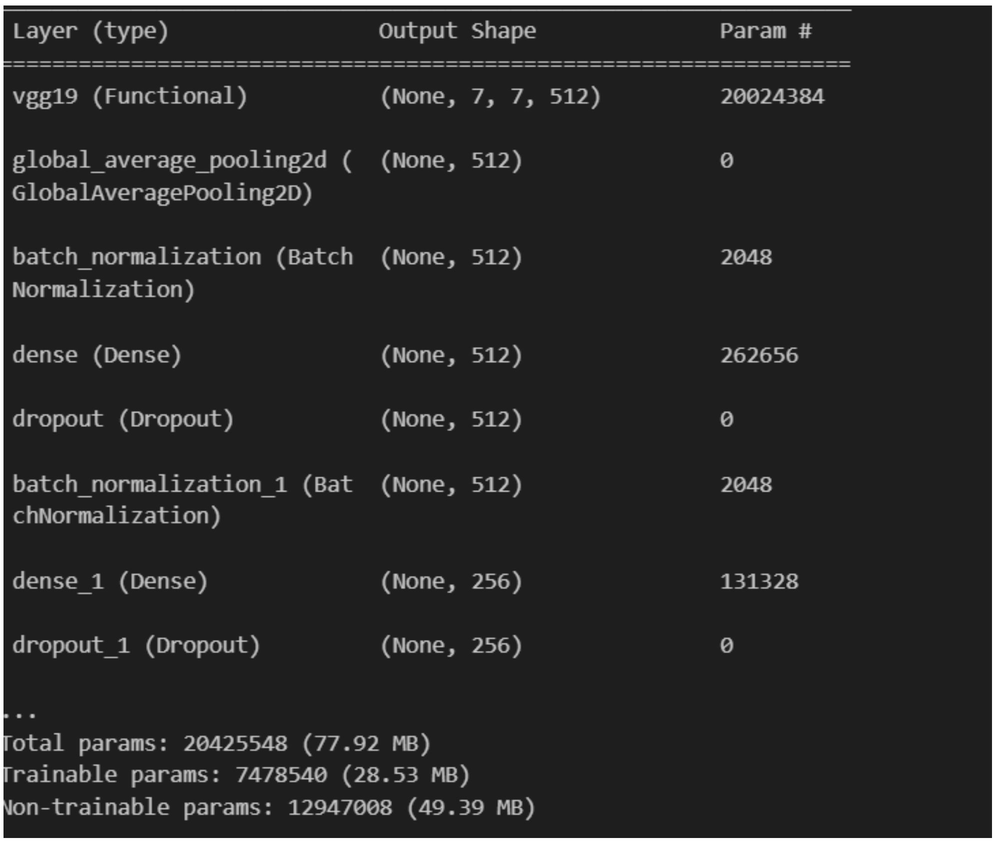
Mean Accuracy: With a mean accuracy of 73.890%, it indicates that, on average, the model correctly classified approximately 73.890% of the instances in the dataset.

Mean Recall: The mean recall of 63.721% suggests that, on average, the model correctly identified approximately 63.721% of the true positive instances among all positive instances in the dataset.

Mean Precision: A mean precision of 65.346% indicates that, on average, approximately 65.346% of the instances predicted as positive by the model were actually true positive instances.

Transfer Learning Network V4

This neural network utilizes the pre-trained VGG19 model, initially trained on ImageNet. The top layers of VGG19 are excluded, allowing for customization to a specific task. After freezing most layers, a new Sequential model extends the architecture. It starts with a GlobalAveragePooling2D layer for feature aggregation, followed by Batch Normalization for stabilization. Two dense layers follow, each with ReLU activation and L2 regularization with a coefficient of 0.001 to prevent overfitting. Dropout layers are then inserted with a dropout rate of 0.6 to further combat overfitting. Finally, the output layer with softmax activation facilitates multi-class classification.



```
 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 7, 7, 512)         20024384

 global_average_pooling2d (  (None, 512)               0
 GlobalAveragePooling2D)

 batch_normalization (Batch  (None, 512)               2048
 Normalization)

 dense (Dense)               (None, 512)               262656

 dropout (Dropout)           (None, 512)               0

 batch_normalization_1 (Bat  (None, 512)               2048
 chNormalization)

 dense_1 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

...
otal params: 20425548 (77.92 MB)
rainable params: 7478540 (28.53 MB)
on-trainable params: 12947008 (49.39 MB)
```

The Data Augmentation is done only on the class ‘Helicopter’ that presents the minor number of samples.

Model Optimization and Compilation - Transfer Learning V4

The model is optimized and compiled using the Adam optimizer with a lower learning rate set

- ***Learning rate:*** 1e-3
- ***Beta 1:*** 0.9
- ***Beta 2:*** 0.999
- ***Epsilon***: 1e-8
- ***AMSGrad***: True
- ***Clipnorm:*** 1.0

- ***Clipvalue:*** 0.5

Adam optimizer is preferred for its robustness in handling sparse gradients and adaptability across various data types. The categorical cross-entropy loss function is employed, given its suitability for multi-class classification tasks. Accuracy is designated as the evaluation metric to gauge the model's performance during both training and validation phases.

Batch Size and Epochs - Transfer Learning Network V4

During training, a batch size of **32** is utilized, meaning that 32 samples are processed simultaneously in each iteration of the training loop. This batch size strikes a balance between computational efficiency and model convergence. The training process is executed over **15 epochs**, determining the number of complete passes through the entire training dataset. This choice of epochs allows the model to sufficiently learn the underlying patterns and features within the data. The number of epochs is caused by the limitation for the **T4 GPU** usage.

Callbacks for Model Training - Transfer Learning Network V4

Several callbacks are integrated to improve the efficiency and stability of the training process:

ModelCheckpoint: This callback monitors validation accuracy (val_accuracy) and saves the model weights to 'model.hdf5' file whenever there's an improvement in performance, ensuring preservation of the best-performing model configuration.

ReduceLROnPlateau: This callback dynamically adjusts the learning rate based on changes in validation loss. If no improvement is observed for a specified number of epochs (patience), set to 3 epochs in this case, the learning rate is reduced by a factor of 0.2. This adaptive learning rate schedule aids in smoother convergence and prevents the model from getting stuck in local minima.

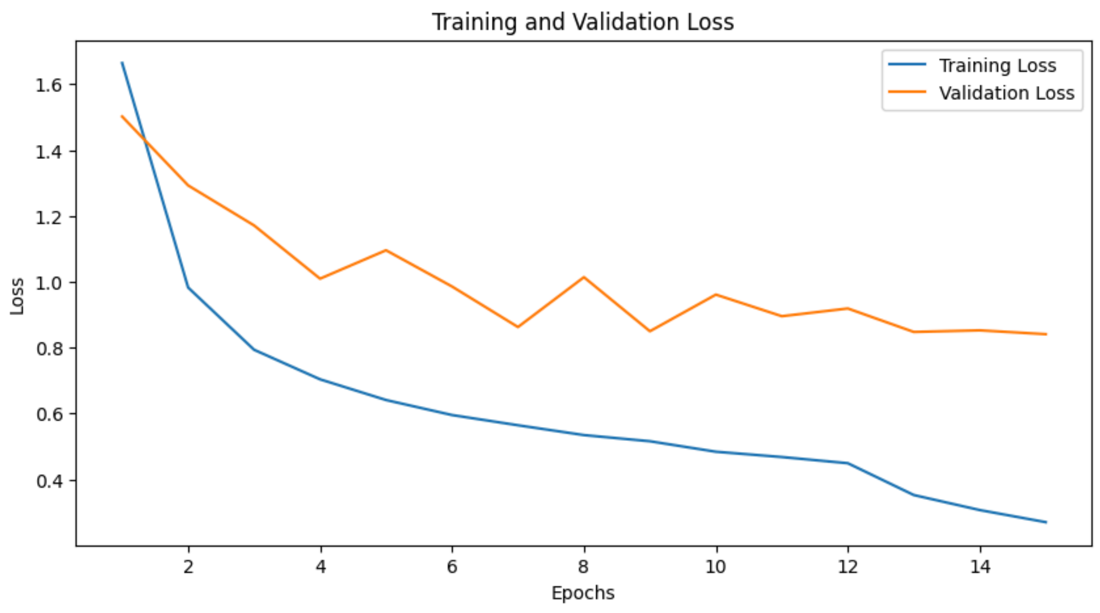
EarlyStopping: This callback terminates the training process if no improvement in validation accuracy is detected for a defined number of epochs, set to 40 epochs. Early stopping prevents overfitting by halting training when further iterations are unlikely to yield significant improvements in generalization performance.

TerminateOnNaN: This callback ensures training halts immediately if any NaN (Not a Number) values are encountered during computation, maintaining the integrity of the training process and preventing invalid calculations or model instability.

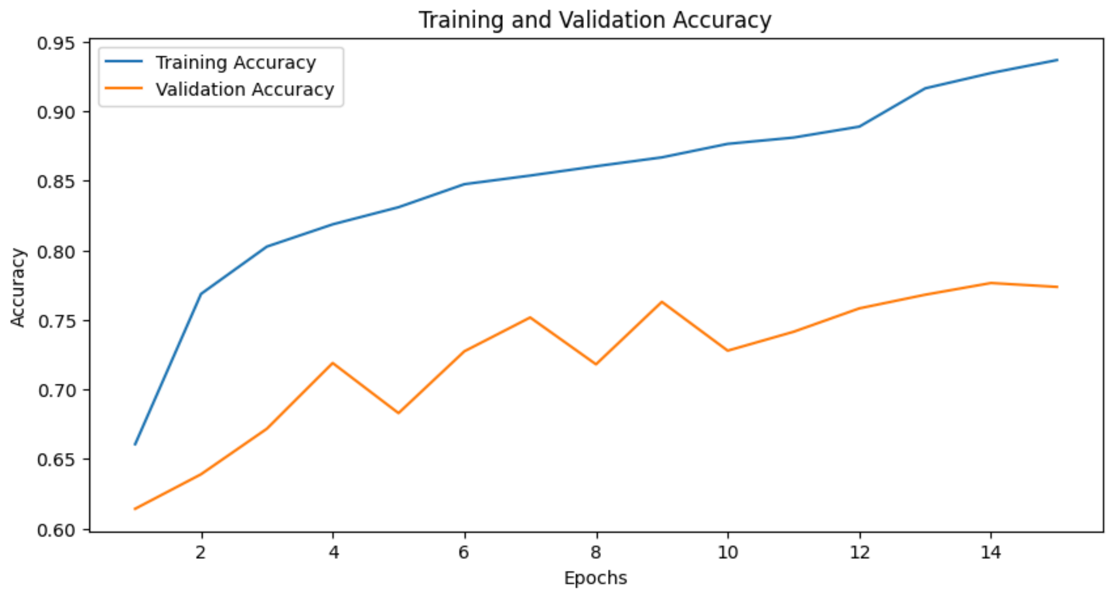
Results - Transfer Learning Network V4

The best validation model was achieved at epoch 14, with a validation accuracy of ***0.7764265537261963***

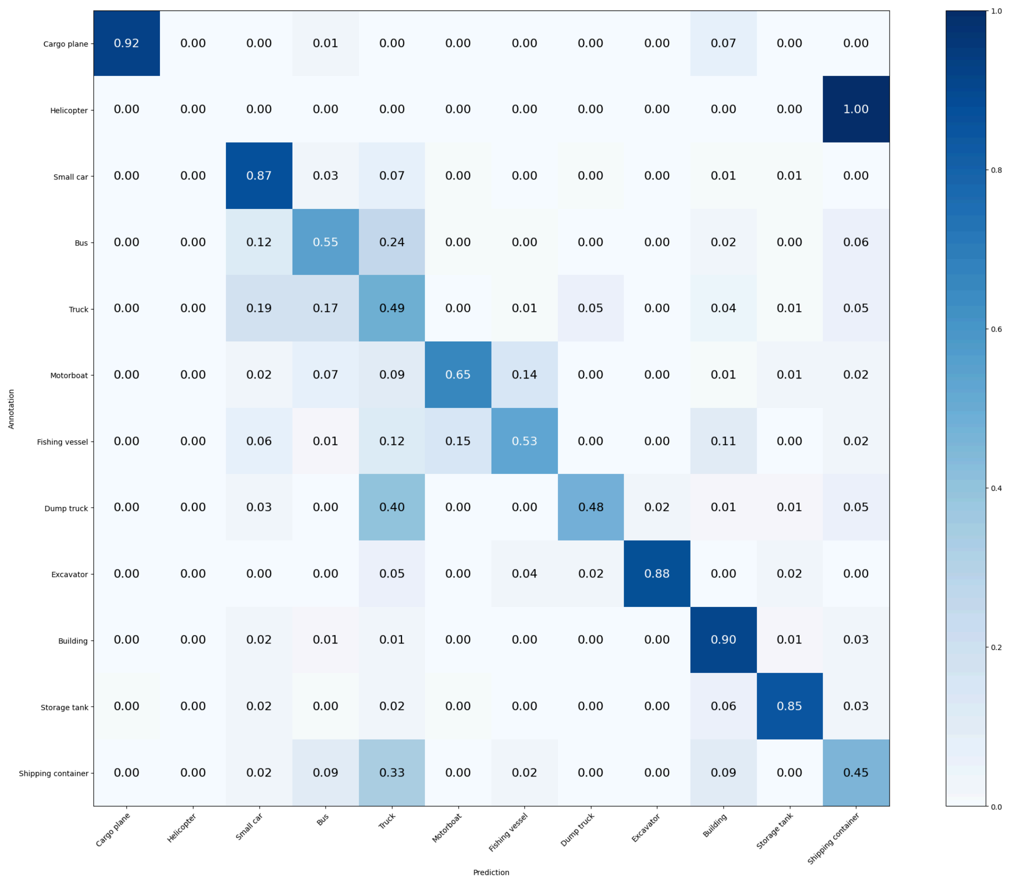
Loss Curves show a really low loss that can be even lower during the epochs but at least is a symptom of overfitting.



The overfitting is proved in the accuracy curves.



The confusion matrix shows good results but the misclassification of the ‘Helicopter’ images is still on. There are some little improvements in class that already worked well with the previous Pre-Trained Network.



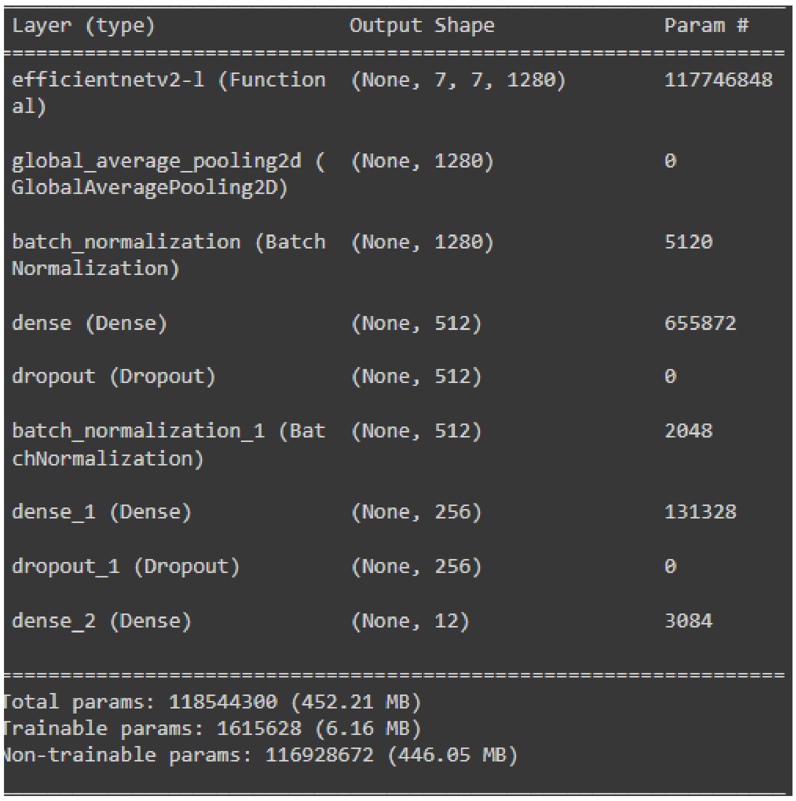
Mean Accuracy: With a mean accuracy of 73.017%, it indicates that, on average, the model correctly classified approximately 73.017% of the instances in the dataset.

Mean Recall: The mean recall of 63.061% suggests that, on average, the model correctly identified approximately 63.061% of the true positive instances among all positive instances in the dataset.

Mean Precision: A mean precision of 65.044% indicates that, on average, approximately 65.044% of the instances predicted as positive by the model were actually true positive instances.

Transfer Learning Network V5

This neural network utilizes the pre-trained EfficientNetV2L model without including the top fully connected layers. The layers of the pre-trained model are frozen except for the last four. A new model is then created on top of the pre-trained model. It adds a GlobalAveragePooling2D layer for feature aggregation, followed by Batch Normalization for stabilization. Then, two dense layers are added, each with ReLU activation and L2 regularization with a coefficient of 0.001 to prevent overfitting. Dropout layers are inserted with a dropout rate of 0.6 to further combat overfitting. Finally, the output layer with softmax activation facilitates multi-class classification.



```
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetv2-l (Function  (None, 7, 7, 1280)        117746848
 al)

 global_average_pooling2d (  (None, 1280)              0
 GlobalAveragePooling2D)

 batch_normalization (Batch  (None, 1280)              5120
 Normalization)

 dense (Dense)               (None, 512)               655872

 dropout (Dropout)           (None, 512)               0

 batch_normalization_1 (Bat  (None, 512)               2048
 chNormalization)

 dense_1 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 12)                3084

=================================================================
otal params: 118544300 (452.21 MB)
rainable params: 1615628 (6.16 MB)
on-trainable params: 116928672 (446.05 MB)
```

Model Optimization and Compilation - Transfer Learning V5

The model is optimized and compiled using the Adam optimizer with the following configurations:

- ***Learning rate***: 1e-3
- ***Beta*** 1: 0.9
- ***Beta*** 2: 0.999
- ***Epsilon***: 1e-8
- ***AMSGrad***: True
- ***Clipnorm***: 1.0
- ***Clipvalue***: 0.5

The Adam optimizer is chosen for its robustness in handling sparse gradients and adaptability across various data types. It utilizes the first and second moments of the gradients to adaptively adjust the learning rates for each parameter. The AMSGrad variant ensures that the adaptive learning rate is monotonically decreasing, which can prevent overshooting during training.

For the loss function, categorical cross-entropy is employed, which is well-suited for multi-class classification tasks. It measures the dissimilarity between the predicted probability distribution and the true distribution of the target classes.

Batch Size and Epochs - Transfer Learning Network V5

During training, a batch size of **32** is utilized, which means that 32 samples are processed simultaneously in each iteration of the training loop. This choice balances computational efficiency and model convergence.
Epochs: The training process is executed over **10 epochs**. Each epoch represents one complete pass through the entire training dataset. This number of epochs allows the model to sufficiently learn the underlying patterns and features within the data. The limitation for **T4 GPU** usage influences the choice of epochs.

Callbacks for Model Training - Transfer Learning Network V5

Several callbacks are integrated to improve the efficiency and stability of the training process:

ModelCheckpoint: This callback monitors validation accuracy (val_accuracy) and saves the model weights to 'model.hdf5' file whenever there's an improvement in performance, ensuring preservation of the best-performing model configuration.

ReduceLROnPlateau: This callback dynamically adjusts the learning rate based on changes in validation loss. If no improvement is observed for a specified number of epochs (patience), set to 3 epochs in this case, the learning rate is reduced by a factor of 0.2. This adaptive learning rate schedule aids in smoother convergence and prevents the model from getting stuck in local minima.

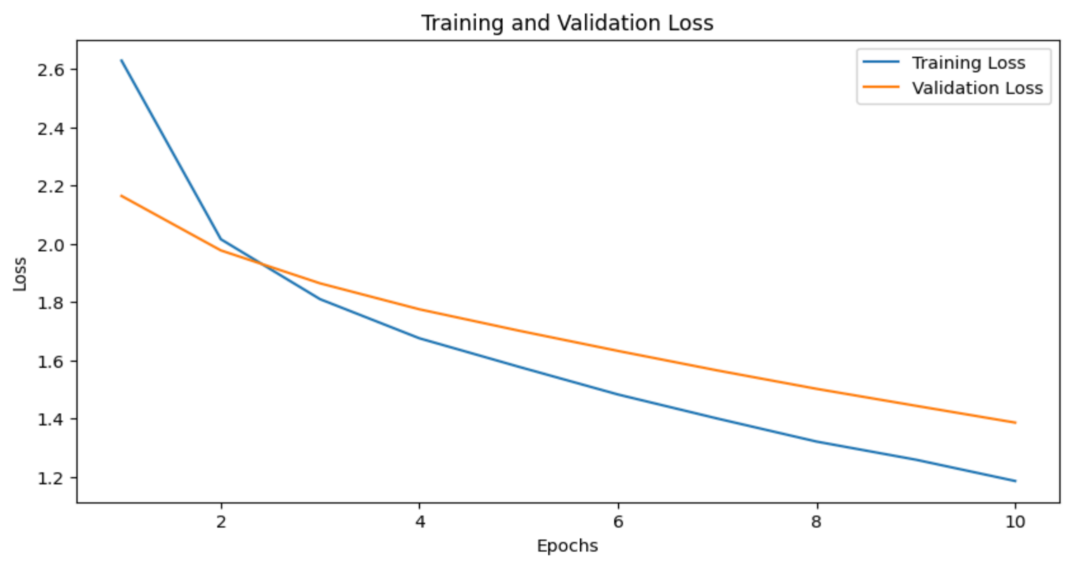
EarlyStopping: This callback terminates the training process if no improvement in validation accuracy is detected for a defined number of epochs, set to 40 epochs. Early stopping prevents overfitting by halting training when further iterations are unlikely to yield significant improvements in generalization performance.

TerminateOnNaN: This callback ensures training halts immediately if any NaN (Not a Number) values are encountered during computation, maintaining the integrity of the training process and preventing invalid calculations or model instability.

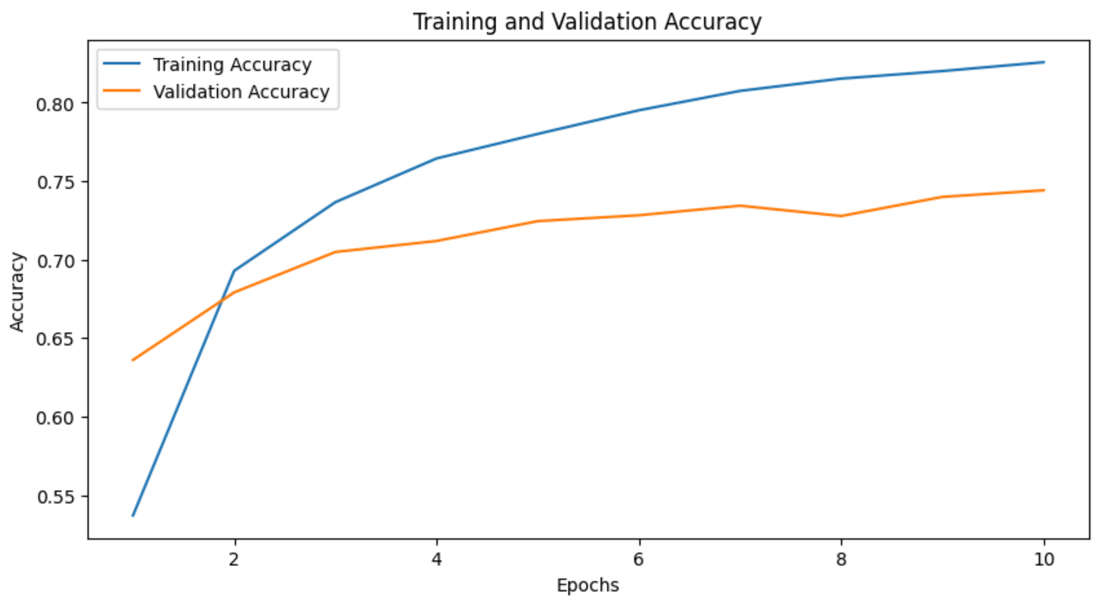
Results - Transfer Learning Network V5

The best validation model was achieved at epoch 10, with a validation accuracy of **0.7441534399986267**.

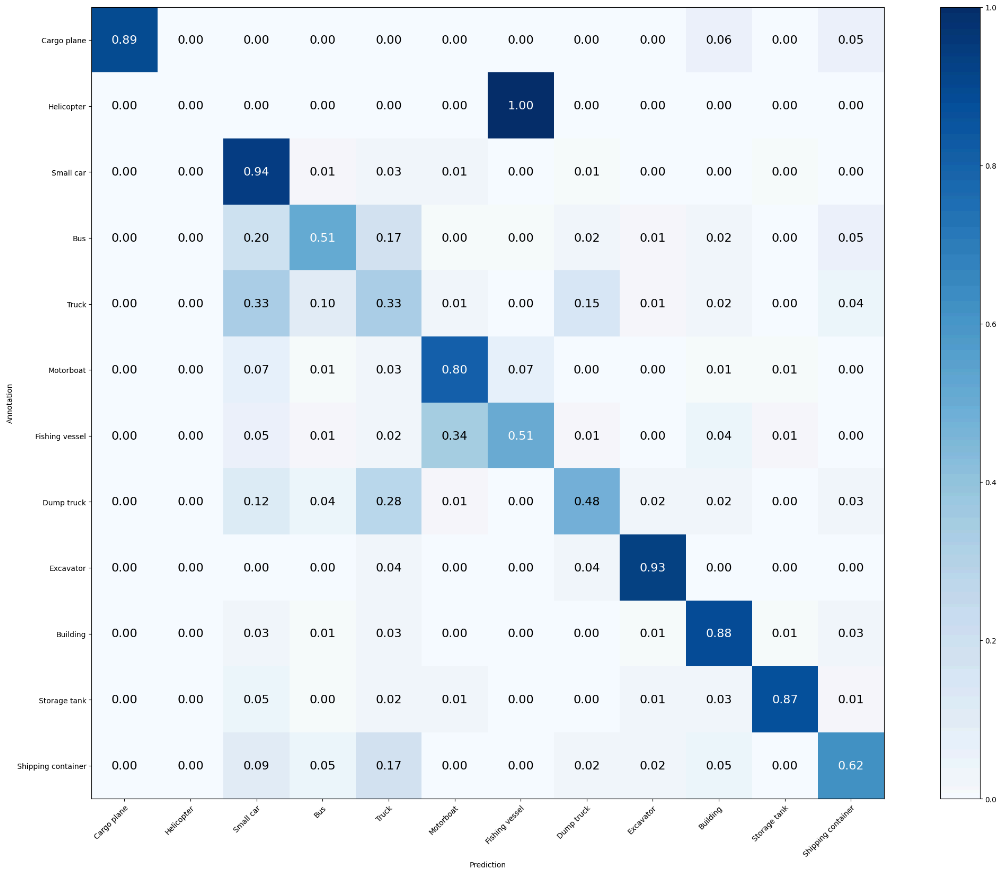
The loss curves are really similar to the ones of the V2 experiment. The gap between Training and Validation loss curves is reduced.



Accuracy curves show a reduced overfitting with respect to other pre-trained models.



The confusion matrix shows the same problems of the other models: the misclassification of the ‘Helicopter’ class. The results are good in almost every class.



Annotation \ Prediction	Cargo plane	Helicopter	Small car	Bus	Truck	Motorboat	Fishing vessel	Dump truck	Excavator	Building	Storage tank	Shipping container
Cargo plane	0.89	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.05
Helicopter	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
Small car	0.00	0.00	0.94	0.01	0.03	0.01	0.00	0.01	0.00	0.00	0.00	0.00
Bus	0.00	0.00	0.20	0.51	0.17	0.00	0.00	0.02	0.01	0.02	0.00	0.05
Truck	0.00	0.00	0.33	0.10	0.33	0.01	0.00	0.15	0.01	0.02	0.00	0.04
Motorboat	0.00	0.00	0.07	0.01	0.03	0.80	0.07	0.00	0.00	0.01	0.01	0.00
Fishing vessel	0.00	0.00	0.05	0.01	0.02	0.34	0.51	0.01	0.00	0.04	0.01	0.00
Dump truck	0.00	0.00	0.12	0.04	0.28	0.01	0.00	0.48	0.02	0.02	0.00	0.03
Excavator	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.04	0.93	0.00	0.00	0.00
Building	0.00	0.00	0.03	0.01	0.03	0.00	0.00	0.00	0.01	0.88	0.01	0.03
Storage tank	0.00	0.00	0.05	0.00	0.02	0.01	0.00	0.00	0.01	0.03	0.87	0.01
Shipping container	0.00	0.00	0.09	0.05	0.17	0.00	0.00	0.02	0.02	0.05	0.00	0.62

Mean Accuracy: With a mean accuracy of 74.421%, it indicates that, on average, the model correctly classified approximately 74.421% of the instances in the dataset.

Mean Recall: The mean recall of 64.687% suggests that, on average, the model correctly identified approximately 64.687% of the true positive instances among all positive instances in the dataset.

Mean Precision: A mean precision of 65.208% indicates that, on average, approximately 65.208% of the instances predicted as positive by the model were actually true positive instances.

Feedback on the used platforms

The predominant platform employed for this project is **Google Colab**. Assessing this platform, the balance between positive and negative aspects skews towards the negative. Positive attributes encompass its user-friendly interface, ease of operation, and efficient collaboration features. However, significant issues arise intermittently, such as unexpected halting of execution, resulting in the loss of all accumulated progress.
Additionally, we attempted to utilize **Kaggle** for this project. However, we encountered obstacles as it presented errors related to installing libraries within the Kaggle Notebook environment, which we were unable to resolve.
In the end, we tried to send our networks to **Cesvima**, through the account that UPM gaved us, but we couldn't manage to complete the setup with Magerit after installing Putty as an SSH client.
Considering all this, we decided to do more experiments using other networks, always using Google Colab.

Conclusion

In the final assignment of the Deep Learning course, we encountered numerous challenges in tackling the image recognition problem while delving into the evolution of deep learning. We commenced our journey with feed-forward Neural Nets (ffNN), traversed through Convolutional Neural Nets (CNNs), and culminated by training popular architectures from scratch using transfer learning. Beginning with dataset analysis, we meticulously evaluated our network performances on the testing dataset, fine-tuning hyperparameters to enhance results. Ultimately, we achieved notable outcomes, experimenting with various networks to enrich our analysis.
Our most promising results were achieved as follows: ffNNs yielded the best performance in the second experiment, CNNs excelled in the initial experiment, and regarding Transfer Learning, the final experiment showcased superior results. Additionally, it's worth noting that the performance across Transfer Learning experiments could potentially have been further improved if not for the stringent computational limitations we faced. Especially, we would have likely addressed the misclassification issue related to the "Helicopter" class had we been afforded more epochs for training in the Transfer Learning experiments. As we did, focusing on data augmentation for a particular class, such as the "Helicopter" class, could have been a strategic decision. This approach involves augmenting the available data for that specific class by applying transformations such as rotation, scaling, or flipping.