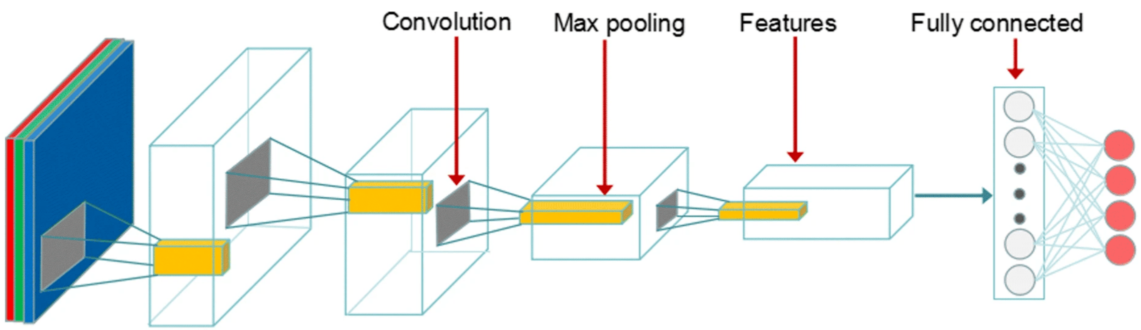
“Bees or Wasp” Dataset Classification using CNN’s with Transfer Learning via Fine-tuning



IMAGENET

Universidad Industrial de Santander

Camilo Andrés Calderón Carrillo 2170090
Jessica Paola Escobar Pérez 2171713
Daniel Felipe Rueda Mariño 2170135

#LaUISqueQueremos

72 años Universidad Industrial de Santander Patrimonio educativo y cultural

Universidad Industrial de Santander

Motivation

Dataset: Bee or Wasp

Figure 1. Dataset classes.

Class	Number of Images	Training Images (80%)	Test Images (10%)	Validation Images (10%)
Bee	3183	2546	318	319
Wasp	4943	3954	494	495
Other Insect	2439	1951	244	244
Other No Insect	856	685	86	85
Total	**11421**	**9136**	**1142**	**1143**

Table 1. Dataset distribution.

Strategies

Strategy	Description
1	Bee, wasp, and insects classes with data augmentation.
2	Bee, wasp, and insects classes without data augmentation.
3	Bee, wasp, insects, and no insects classes with data augmentation.
4	Bee, wasp, insects, and no insects classes without data augmentation.

Table 2. Strategies.

Data Augmentation

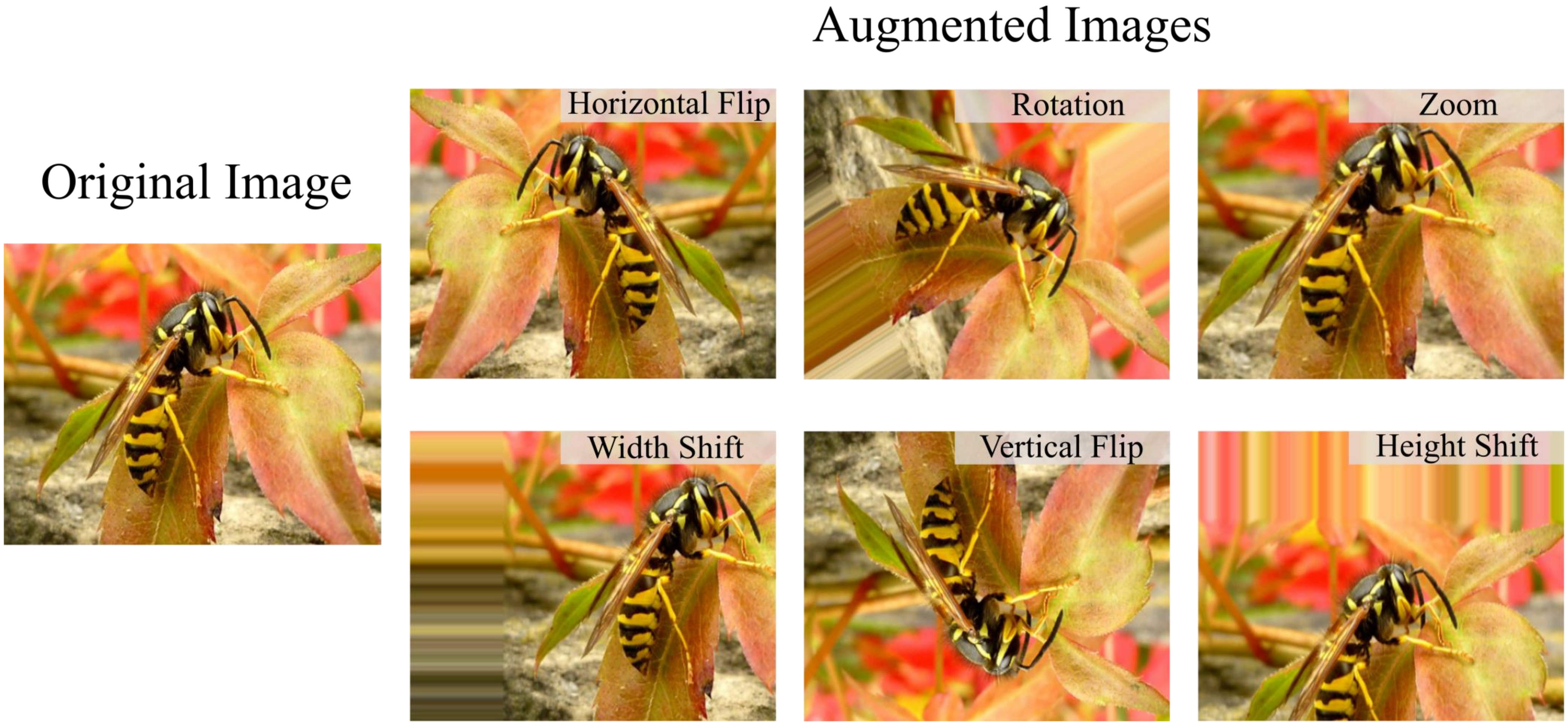


Figure 2. Data augmentation.

Data Augmentation

Class	Training images	Augmented images
Bee	2546	**3986**
Wasp	3954	**3986**
Other Insect	1951	**3998**
Other No Insect	685	**3905**

Table 3. Training data augmentation.

Class	Training Images	Test Images	Validation Images
Bee	**3986**	318	319
Wasp	**3986**	494	495
Other Insect	**3998**	244	244
Other No Insect	**3905**	86	85

Table 4. Dataset augmented.

Strategy 1: Total Training images: 11970 Total images: **14084**

Strategy 3: Total Training images: 15875 Total images: **18160**

Model Choice

For the model choice, we tested with the original data, i.e., four classes without data augmentation.

Model	Accuracy	Precision	Recall	F1-Score	SPEC
DenseNet201	0.9186	0.9187	0.9186	0.9183	0.9707
ResNet50	**0.9440**	**0.9451**	**0.9440**	**0.9437**	**0.9805**
VGG16	0.9204	0.9209	0.9204	0.9201	0.9673
VGG19	0.9186	0.9185	0.9186	0.9181	0.9646

Table 5. Results.

Model: ResNet50

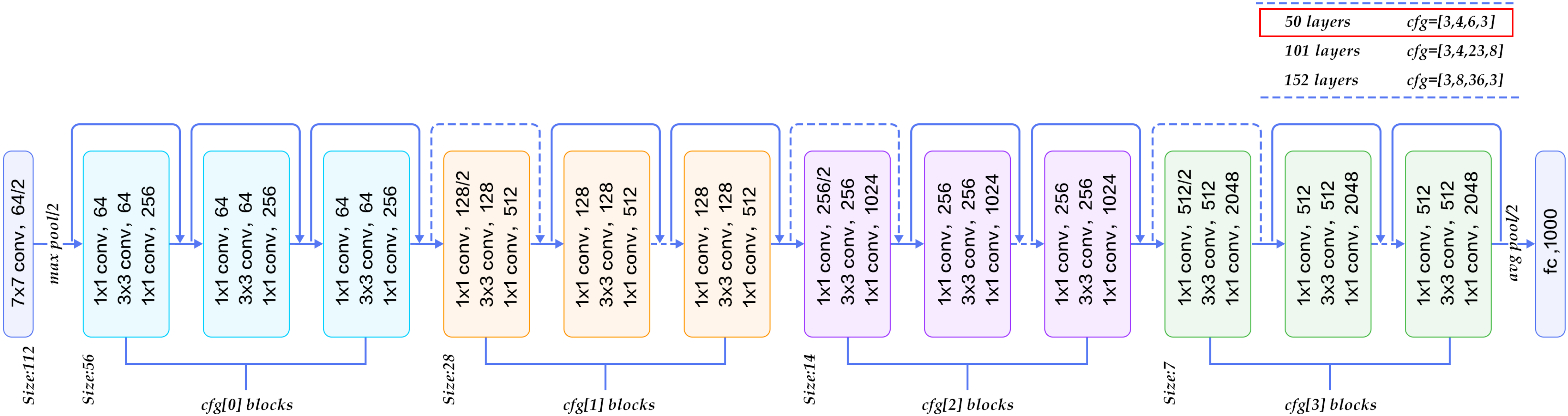


Figure 3. ResNet50.

Transfer Learning via Fine-Tuning


```
#LOADING THE RESNET50 MODEL WITH IMAGENET PRE-TRAINED WEIGHTS
base_model = tf.keras.applications.ResNet50(
    include_top=False, weights='imagenet', input_tensor=None, input_shape=(224,224,3),
     classes=4, pooling='max', classifier_activation='softmax')

base_model.trainable = True
set_trainable = False

#FINE-TUNING
for layer in base_model.layers:
  if layer.name == 'conv4_block4_3_conv':
    set_trainable = True
  if set_trainable:
    layer.trainable = True
  else:
    layer.trainable = False

base_model.summary()
```

Figure 4. Transfer learning.

Model Layers and Hyperparameters

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 2048)              23587712
_________________________________________________________________
dropout (Dropout)            (None, 2048)              0
_________________________________________________________________
batch_normalization (BatchNo (None, 2048)              8192
_________________________________________________________________
dense (Dense)                (None, 256)               524544
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 1028
=================================================================
Total params: 24,122,500
Trainable params: 18,008,836
Non-trainable params: 6,113,664
_________________________________________________________________
```

Figure 5. Model - 4 classes.

Hyperparameters	
Optimizer	Adam
Learning rate	0,001
Epochs	20
Batch size	32
Image shape	(224,224,3)

Table 6. Hyperparameters.

Model Layers and Hyperparameters

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 2048)              23587712
_________________________________________________________________
dropout (Dropout)            (None, 2048)              0
_________________________________________________________________
batch_normalization (BatchNo (None, 2048)              8192
_________________________________________________________________
dense (Dense)                (None, 256)               524544
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 771
=================================================================
Total params: 24,122,243
Trainable params: 18,008,579
Non-trainable params: 6,113,664
_________________________________________________________________
```

Figure 6. Model – 3 classes.

Hyperparameters	
Optimizer	Adam
Learning rate	0,001
Epochs	20
Batch size	32
Image shape	(224,224,3)

Table 6. Hyperparameters.

Training Parameters

```
#TRAINING PARAMETERS
filepath = "model.h5"

checkpoint_param = {
    "filepath": filepath,
    "monitor": "val_categorical_accuracy",
    "verbose": 1,
    "save_best_only": True,
    "mode": "max"
}
checkpoint = ModelCheckpoint(**checkpoint_param)

lr_decay_params = {
    "monitor": "val_loss",
    "factor": 0.5,
    "patience": 2,
    "min_lr": 1e-5
}
lr_decay = ReduceLROnPlateau(**lr_decay_params)
```

Figure 7. Callbacks.

```
#TRAINING
fit_params = {
    "generator": train_flow,
    "steps_per_epoch": train_flow.n // batch_size,
    "epochs": 20,
    "verbose": 1,
    "validation_data": val_flow,
    "validation_steps": val_flow.n // batch_size,
    "callbacks": [checkpoint, lr_decay]
}
print("Training the model...")

history = model.fit_generator(**fit_params)

print("Done!")
```

Figure 8. Training.

Training Parameters

```
#TRAINING PARAMETERS
filepath = "model.h5"

checkpoint_param = {
    "filepath": filepath,
    "monitor": "val_categorical_accuracy",
    "verbose": 1,
    "save_best_only": True,
    "mode": "max"
}
checkpoint = ModelCheckpoint(**checkpoint_param)

lr_decay_params = {
    "monitor": "val_loss",
    "factor": 0.5,
    "patience": 2,
    "min_lr": 1e-5
}
lr_decay = ReduceLROnPlateau(**lr_decay_params)
```

Figure 7. Callbacks.

```
#TRAINING
fit_params = {
    "generator": train_flow,
    "steps_per_epoch": train_flow.n // batch_size,
    "epochs": 20,
    "verbose": 1,
    "validation_data": val_flow,
    "validation_steps": val_flow.n // batch_size,
    "callbacks": [checkpoint, lr_decay]
}
print("Training the model...")

history = model.fit_generator(**fit_params)

print("Done!")
```

Figure 8. Training.

Strategy Choice

For the strategy choice, we tested with the four strategies, and the results are shown below:

Strategy	Accuracy	Precision	Recall	F1-Score	SPEC
1	**0.9508**	**0.9507**	**0.9508**	**0.9508**	**0.9731**
2	0.9461	0.9460	0.9461	0.9457	0.9696
3	**0.9458**	**0.9460**	**0.9458**	**0.9456**	**0.9834**
4	0.9440	0.9451	0.9440	0.9437	0.9805

Table 7. Results.

1. Bee, wasp, and insects classes with data augmentation.
2. Bee, wasp, and insects classes without data augmentation.
3. Bee, wasp, insects, and no insects classes with data augmentation.
4. Bee, wasp, insects, and no insects classes without data augmentation.

Results (Strategy 1)

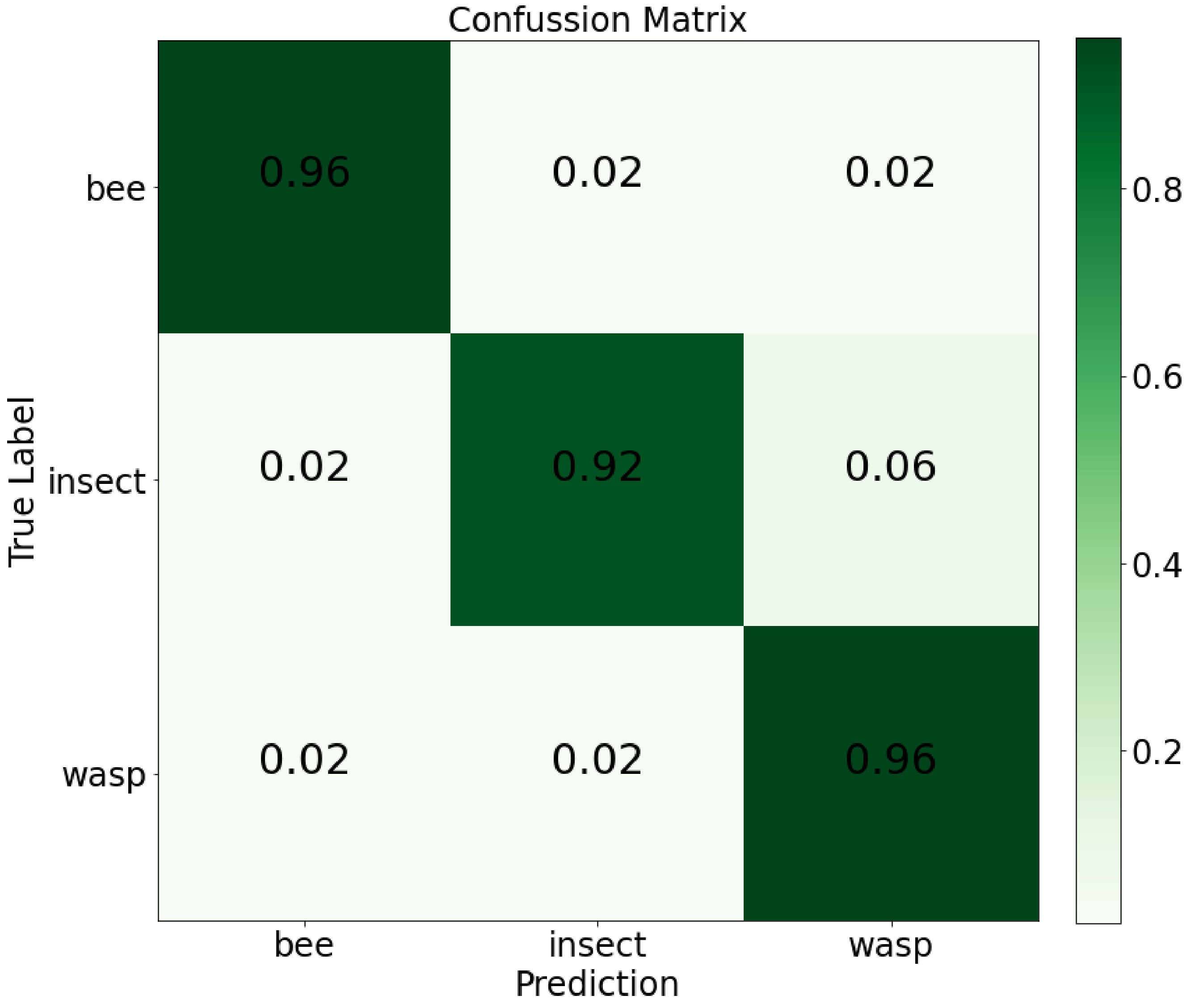


Figure 9. Confussion matrix.

Results (Strategy 1)

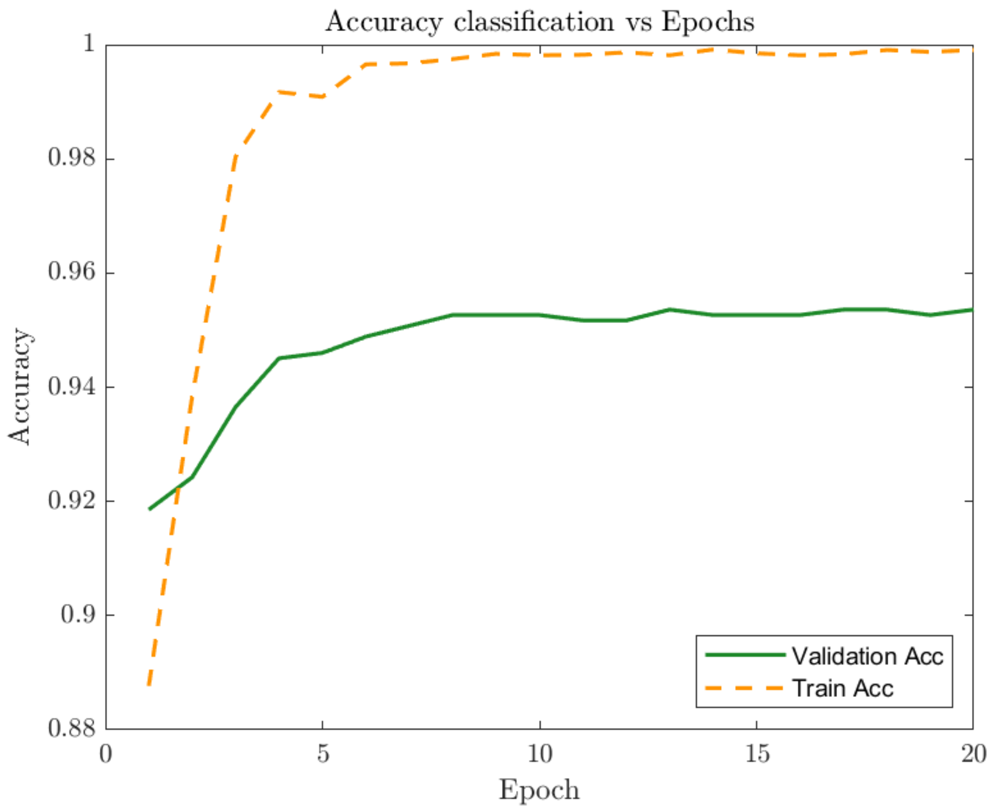


Figure 10. Accuracy vs Epochs.

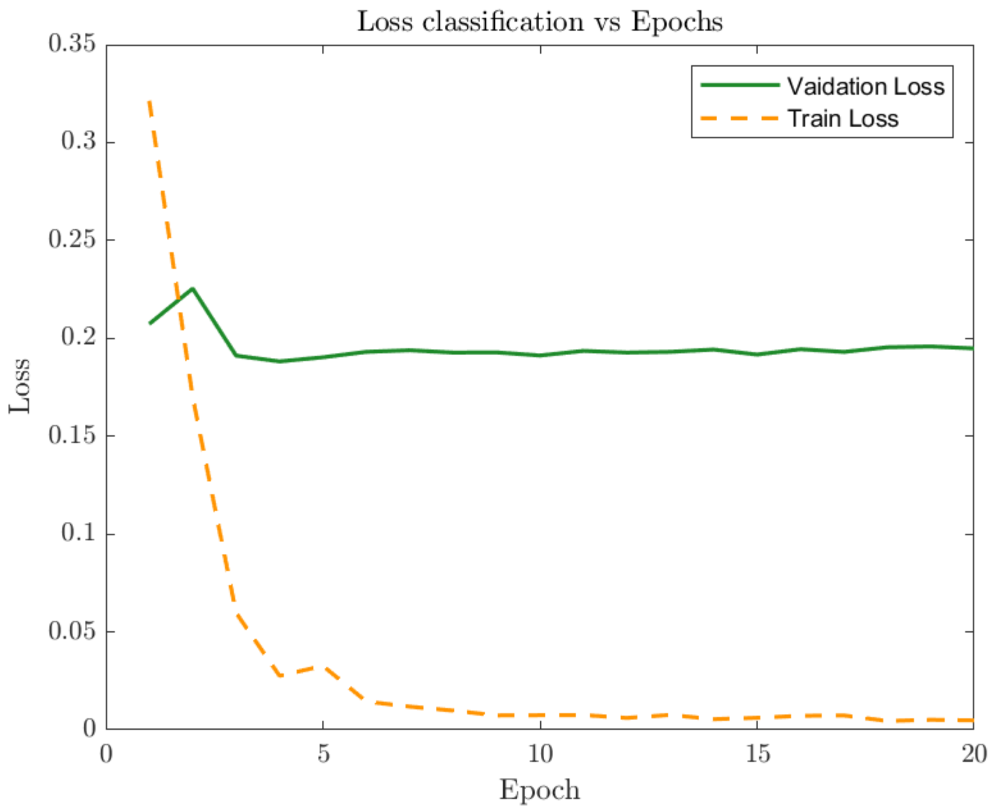


Figure 11. Loss vs Epochs.

Results (Strategy 1)

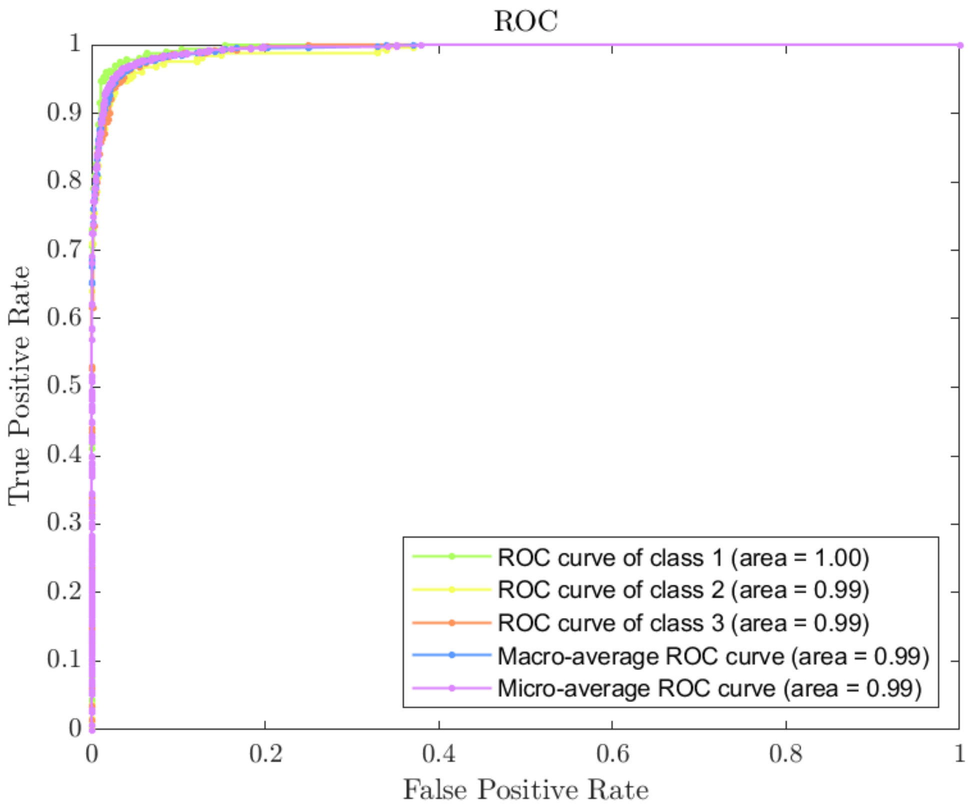


Figure 12. ROC curve.

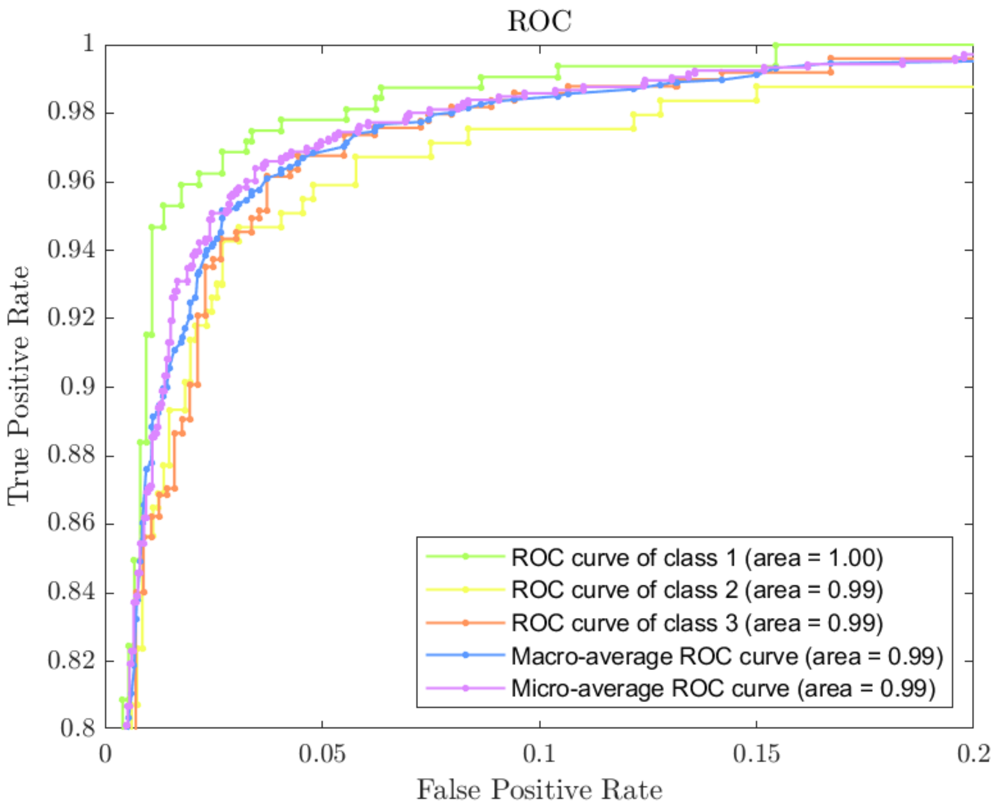


Figure 13. ROC curve - zoom.

Results (Strategy 3)

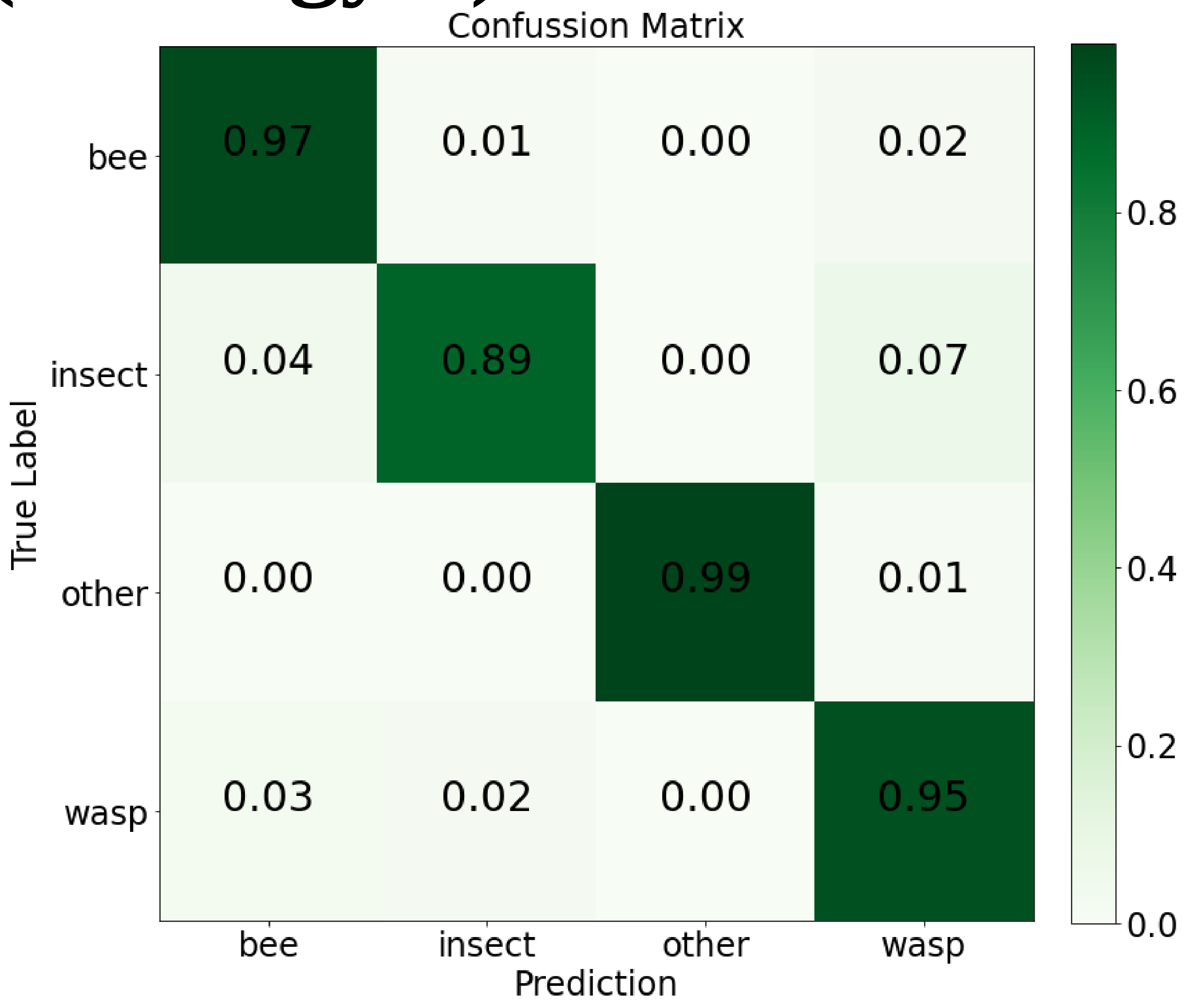


Figure 14. Confussion matrix.

Universidad Industrial de Santander

Somos **el mejor** escenario de creación e innovación.

www.uis.edu.co

Results (Strategy 3)

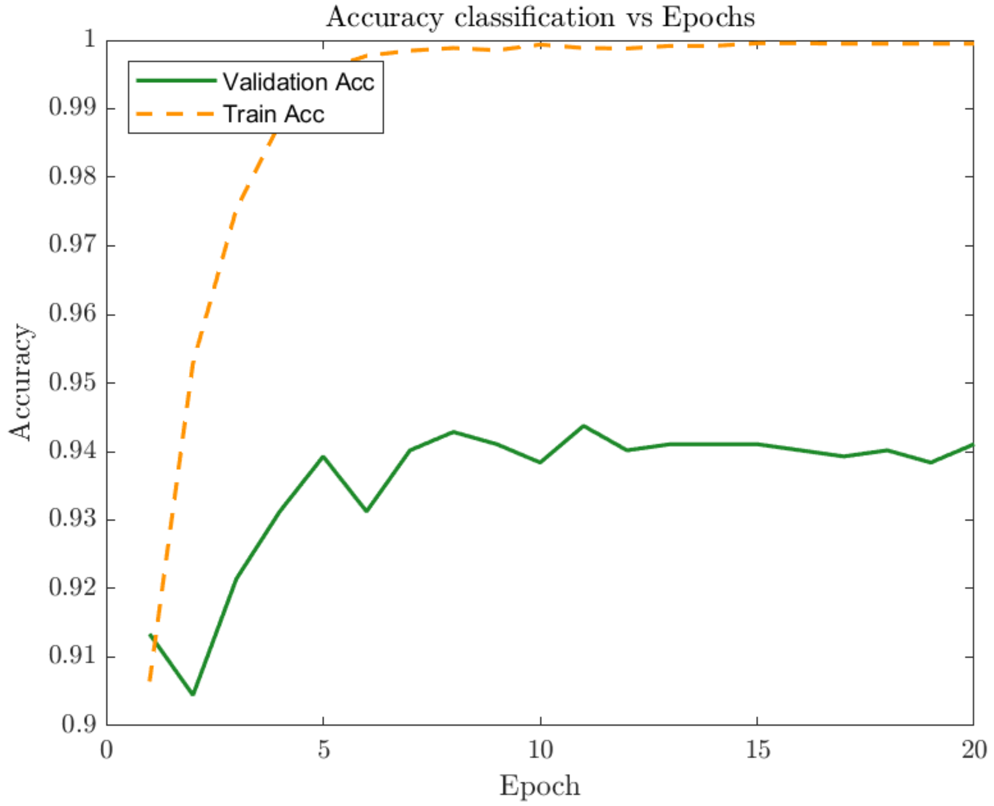


Figure 15. Accuracy vs Epochs.

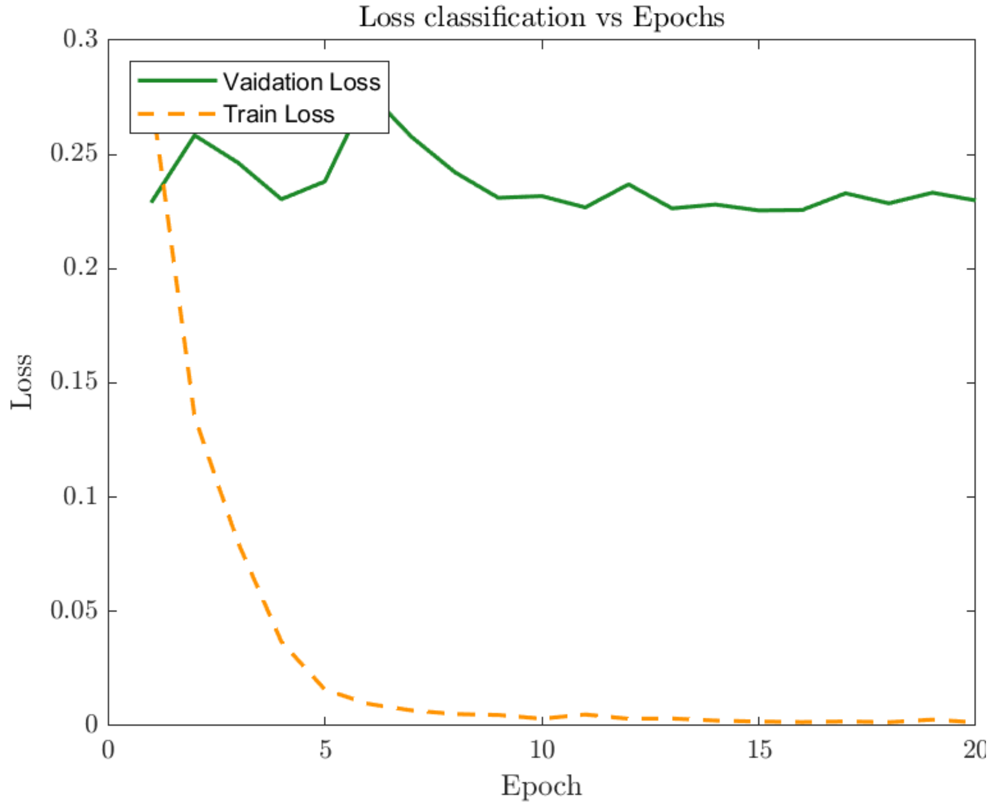


Figure 16. Loss vs Epochs.

Universidad Industrial de Santander

Somos el mejor escenario de creación e innovación.
www.uis.edu.co

Results (Strategy 3)

Universidad Industrial de Santander

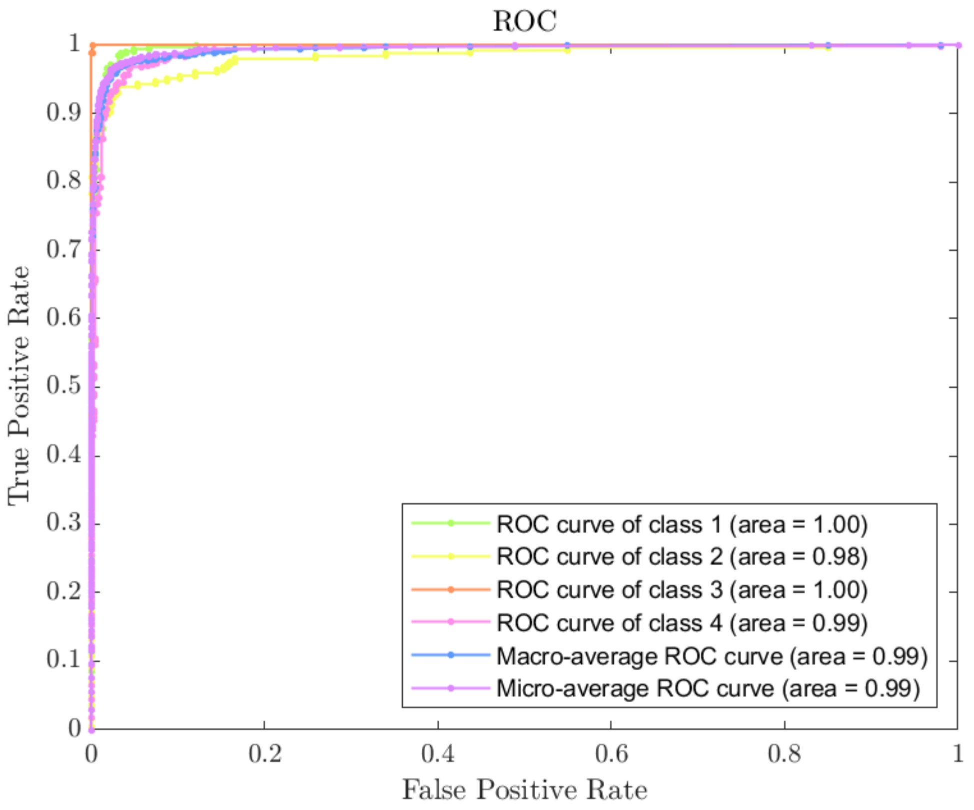


Figure 17. ROC curve.

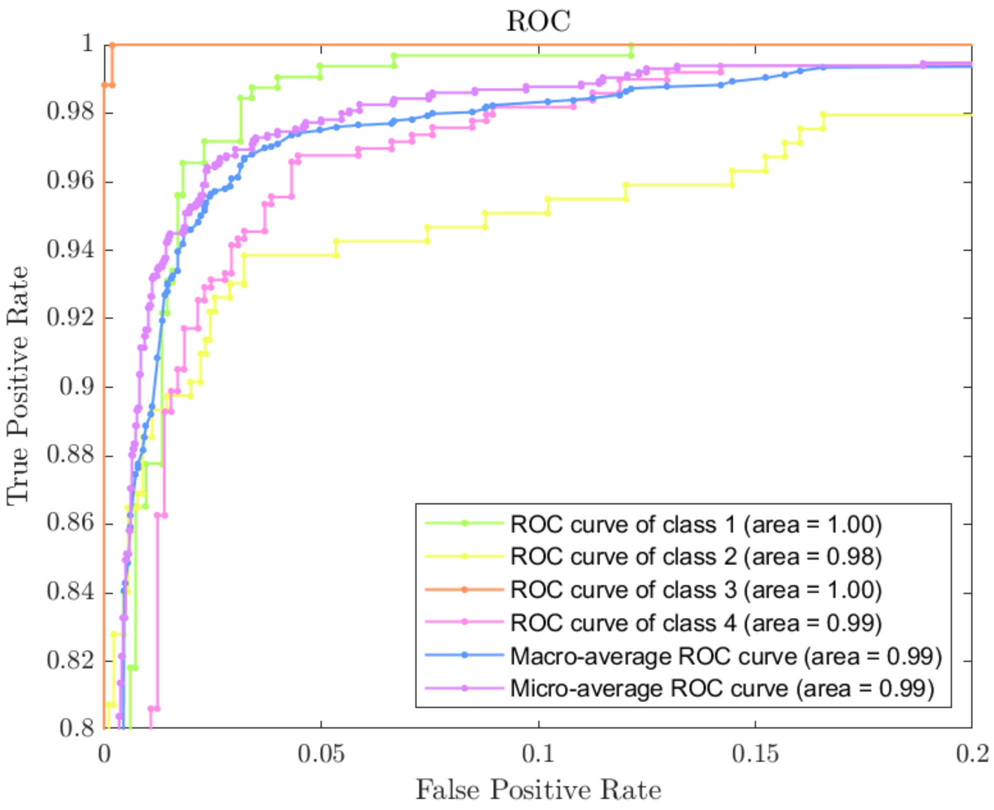


Figure 18. ROC curve - zoom.

Conclusions

- We proposed a method using ResNet50 Convolutional Neural Network over “Bees vs Wasps” dataset combining techniques such as transfer learning, fine-tuning, and data augmentation which can increases remarkably the model performance.

Universidad Industrial de Santander

#LaUISqueQueremos

¡Thanks!