Practice 1: Convolutional Neural Network - Deep Learning course

- Alejandro Dopico Castro (alejandro.dopico2@udc.es).
- Ana Xiangning Pereira Ezquerro (ana.ezquerro@udc.es).

In [ ]:
```
import plotly
plotly.offline.init_notebook_mode()
```

In [ ]:
```
import tensorflow as tf
from keras.optimizers import Adam
from keras.metrics import SparseCategoricalAccuracy
from logging import ERROR
tf.get_logger().setLevel(ERROR)
import warnings, os
import plotly.express as px
warnings.filterwarnings("ignore")
from utils import *
from models import *
from typing import List, Tuple
model_accuracies: List[Tuple[str, int]] = []

# global variables
IMG_SIZE = 100
BATCH_SIZE = 258
base_dir = 'animals/'
model_dir = 'results/'
if not os.path.exists(model_dir):
    os.mkdir(model_dir)
```

```
2024-03-07 11:56:17.619575: I tensorflow/core/util/port.cc:113] oneDNN cus
tom operations are on. You may see slightly different numerical results du
e to floating-point round-off errors from different computation orders. To
turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-03-07 11:56:17.640942: E external/local_xla/xla/stream_executor/cuda/
cuda_dnn.cc:9261] Unable to register cuDNN factory: Attempting to register
factory for plugin cuDNN when one has already been registered
2024-03-07 11:56:17.640961: E external/local_xla/xla/stream_executor/cuda/
cuda_fft.cc:607] Unable to register cuFFT factory: Attempting to register
factory for plugin cuFFT when one has already been registered
2024-03-07 11:56:17.641561: E external/local_xla/xla/stream_executor/cuda/
cuda_blas.cc:1515] Unable to register cuBLAS factory: Attempting to regist
er factory for plugin cuBLAS when one has already been registered
2024-03-07 11:56:17.645228: I tensorflow/core/platform/cpu_feature_guard.c
c:182] This TensorFlow binary is optimized to use available CPU instructio
ns in performance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operatio
ns, rebuild TensorFlow with the appropriate compiler flags.
2024-03-07 11:56:18.014887: W tensorflow/compiler/tf2tensorrt/utils/py_uti
ls.cc:38] TF-TRT Warning: Could not find TensorRT
```

Note: Some figures of this notebook have been rendered using Plotly. Depending on the server the notebook is executed in, the Plotly renderer should be modified to properly display the figures:

In [ ]:
```
import plotly.io as pio
pio.renderers.default = 'vscode' # In our case, we used visual studio cod
```

Table of Contents

Exercise 1. Dataset preprocessing

We decided to use for the image resolution 100×100 for retrieving similar results than considering higher resolutions with the simpler architectures. The batch size is adapted to fit the GPU capabilities of the local machine. We used the original validation set for evaluation a split a 15% of the data for validation. All the input images were normalized to fit the range $[0, 1]$ with the Rescaling layer.

In [ ]:
```
train_dataset, val_dataset, test_dataset = load_data(base_dir, IMG_SIZE,
```

```
Found 13474 files belonging to 5 classes.
Using 11453 files for training.
Using 2021 files for validation.
```

```
2024-03-07 11:56:18.884797: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-03-07 11:56:18.906855: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-03-07 11:56:18.906967: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-03-07 11:56:18.907937: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-03-07 11:56:18.908009: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-03-07 11:56:18.908051: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-03-07 11:56:18.957630: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-03-07 11:56:18.957716: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-03-07 11:56:18.957767: I external/local_xla/xla/stream_executor/cuda/
cuda_executor.cc:901] successful NUMA node read from SysFS had negative va
lue (-1), but there must be at least one NUMA node, so returning NUMA node
zero. See more at https://github.com/torvalds/linux/blob/v6.0/Documentatio
n/ABI/testing/sysfs-bus-pci#L344-L355
2024-03-07 11:56:18.957810: I tensorflow/core/common_runtime/gpu/gpu_devic
e.cc:1929] Created device /job:localhost/replica:0/task:0/device:GPU:0 wit
h 22146 MB memory:  -> device: 0, name: NVIDIA GeForce RTX 3090 Ti, pci bu
s id: 0000:01:00.0, compute capability: 8.6
2024-03-07 11:56:19.194639: I external/local_tsl/tsl/platform/default/subp
rocess.cc:304] Start cannot spawn child process: No such file or directory
Found 1497 files belonging to 5 classes.
```

Exercise 2. Custom Convolutional Models

For this exercise, we prepared four different convolutional architectures with increasing complexity and regularization methods to tackle the animal classification problem: (1) a simple linear approach with 2 convolutional layers interleaved with maxpooling, (2) a more complex convolutional-linear model with 3 blocks of paired convolutions and max-poolings, (3) a convolutional architecture based on residual connections (He et al., 2015), (4) the integration of the Inception block (Szegedy et al., 2014). We used the validation split to tune each hyperparameter configuration (although not all experiments are included in this notebook to facilitate the readibility) and applied some regularization methods learnt in previous practical lessons to avoid overfitting.

2.1. Simple Model

In [ ]:

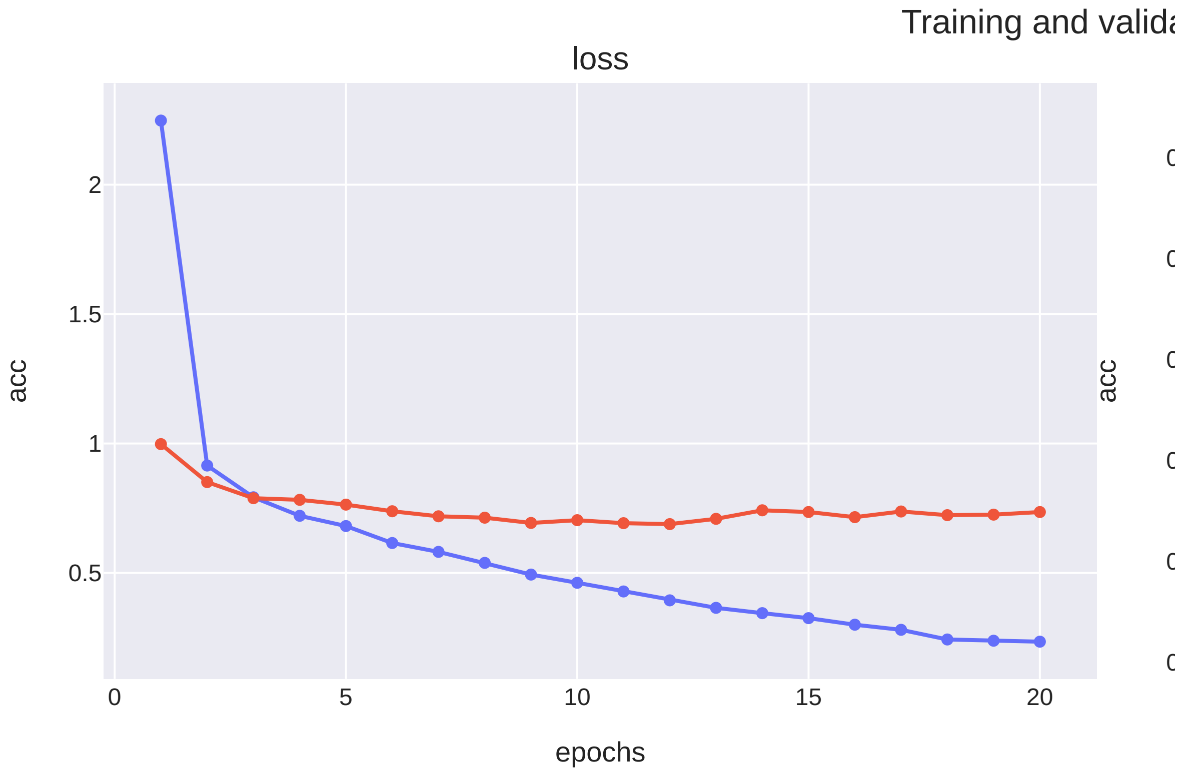
```
simple_model = SimpleModel(num_classes=5)
optimizer = Adam(learning_rate=1e-3)
simple_model.compile(optimizer=optimizer, loss="sparse_categorical_crosse
simple_model_history = train_model(simple_model, train_dataset, val_datas
print(
    f"Final loss: training -> {simple_model_history['loss'][-1]:.2f}, val
)
print(
    f"Final accuracy: training -> {simple_model_history['acc'][-1]:.2f},
        validation -> {simple_model_history['val_acc'][-1]:.2f}"
)
```

```
Epoch 1/20
2024-03-07 11:56:20.945884: I external/local_xla/xla/stream_executor/cuda/
cuda_dnn.cc:454] Loaded cuDNN version 8904
2024-03-07 11:56:21.015085: I external/local_tsl/tsl/platform/default/subp
rocess.cc:304] Start cannot spawn child process: No such file or directory
2024-03-07 11:56:21.395381: I external/local_xla/xla/service/service.cc:16
8] XLA service 0x7fcc386be2a0 initialized for platform CUDA (this does not
guarantee that XLA will be used). Devices:
2024-03-07 11:56:21.395397: I external/local_xla/xla/service/service.cc:17
6]   StreamExecutor device (0): NVIDIA GeForce RTX 3090 Ti, Compute Capabi
lity 8.6
2024-03-07 11:56:21.398714: I tensorflow/compiler/mlir/tensorflow/utils/du
mp_mlir_util.cc:269] disabling MLIR crash reproducer, set env var `MLIR_CR
ASH_REPRODUCER_DIRECTORY` to enable.
WARNING: All log messages before absl::InitializeLog() is called are writt
en to STDERR
I0000 00:00:1709808981.458911  222545 device_compiler.h:186] Compiled clus
ter using XLA!  This line is logged at most once for the lifetime of the p
rocess.
```

```
45/45 [==============================] - 8s 112ms/step - loss: 2.2477 - ac
c: 0.4202 - val_loss: 0.9977 - val_acc: 0.6378 - lr: 0.0010
Epoch 2/20
45/45 [==============================] - 5s 84ms/step - loss: 0.9149 - acc
: 0.6693 - val_loss: 0.8510 - val_acc: 0.6784 - lr: 0.0010
Epoch 3/20
45/45 [==============================] - 5s 87ms/step - loss: 0.7922 - acc
: 0.7127 - val_loss: 0.7887 - val_acc: 0.7006 - lr: 0.0010
Epoch 4/20
45/45 [==============================] - 5s 85ms/step - loss: 0.7207 - acc
: 0.7392 - val_loss: 0.7825 - val_acc: 0.6972 - lr: 0.0010
Epoch 5/20
45/45 [==============================] - 5s 86ms/step - loss: 0.6811 - acc
: 0.7508 - val_loss: 0.7639 - val_acc: 0.7180 - lr: 0.0010
Epoch 6/20
45/45 [==============================] - 5s 86ms/step - loss: 0.6156 - acc
: 0.7795 - val_loss: 0.7383 - val_acc: 0.7229 - lr: 0.0010
Epoch 7/20
45/45 [==============================] - 5s 86ms/step - loss: 0.5814 - acc
: 0.7924 - val_loss: 0.7188 - val_acc: 0.7303 - lr: 0.0010
Epoch 8/20
45/45 [==============================] - 5s 85ms/step - loss: 0.5388 - acc
: 0.8108 - val_loss: 0.7135 - val_acc: 0.7338 - lr: 0.0010
Epoch 9/20
45/45 [==============================] - 5s 87ms/step - loss: 0.4936 - acc
: 0.8272 - val_loss: 0.6931 - val_acc: 0.7378 - lr: 0.0010
Epoch 10/20
45/45 [==============================] - 5s 84ms/step - loss: 0.4619 - acc
: 0.8441 - val_loss: 0.7039 - val_acc: 0.7353 - lr: 0.0010
Epoch 11/20
45/45 [==============================] - 5s 87ms/step - loss: 0.4283 - acc
: 0.8558 - val_loss: 0.6922 - val_acc: 0.7467 - lr: 0.0010
Epoch 12/20
45/45 [==============================] - 5s 87ms/step - loss: 0.3941 - acc
: 0.8707 - val_loss: 0.6886 - val_acc: 0.7481 - lr: 0.0010
Epoch 13/20
45/45 [==============================] - 5s 86ms/step - loss: 0.3651 - acc
: 0.8874 - val_loss: 0.7090 - val_acc: 0.7392 - lr: 0.0010
Epoch 14/20
45/45 [==============================] - 5s 84ms/step - loss: 0.3446 - acc
: 0.8900 - val_loss: 0.7420 - val_acc: 0.7234 - lr: 0.0010
Epoch 15/20
45/45 [==============================] - 5s 83ms/step - loss: 0.3253 - acc
: 0.9008 - val_loss: 0.7353 - val_acc: 0.7373 - lr: 0.0010
Epoch 16/20
45/45 [==============================] - 5s 84ms/step - loss: 0.3000 - acc
: 0.9060 - val_loss: 0.7155 - val_acc: 0.7496 - lr: 0.0010
Epoch 17/20
45/45 [==============================] - 5s 85ms/step - loss: 0.2802 - acc
: 0.9150 - val_loss: 0.7373 - val_acc: 0.7402 - lr: 0.0010
Epoch 18/20
45/45 [==============================] - 5s 83ms/step - loss: 0.2429 - acc
: 0.9347 - val_loss: 0.7233 - val_acc: 0.7427 - lr: 2.0000e-04
Epoch 19/20
45/45 [==============================] - 5s 83ms/step - loss: 0.2381 - acc
: 0.9367 - val_loss: 0.7252 - val_acc: 0.7407 - lr: 2.0000e-04
Epoch 20/20
45/45 [==============================] - 5s 84ms/step - loss: 0.2344 - acc
: 0.9364 - val_loss: 0.7352 - val_acc: 0.7397 - lr: 2.0000e-04
Final loss: training -> 0.23, validation -> 0.74
```

```
Final accuracy: training -> 0.94,        validation -> 0.74
```

In [ ]:
```
plot_history(simple_model_history, ['loss', 'acc'], name='Simple Model')
```



In [ ]:
```
test_loss, test_accuracy = simple_model.evaluate(test_dataset, batch_size
print(
    f"Model evaluated: Test Loss-> {test_loss}, Test Accuracy -> {test_ac
)
print(simple_model.summary())
model_accuracies.append(("SimpleModel", test_accuracy))
```

```
6/6 [==============================] - 1s 68ms/step - loss: 0.6367 - acc:
0.7695
Model evaluated: Test Loss-> 0.6366969347000122, Test Accuracy -> 76.95%
Model: "SimpleModel"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       multiple                  0

 conv2d (Conv2D)             multiple                  896

 max_pooling2d (MaxPooling2  multiple                  0
 D)

 conv2d_1 (Conv2D)           multiple                  18496

 max_pooling2d_1 (MaxPoolin  multiple                  0 (unused)
 g2D)

 flatten (Flatten)           multiple                  0

 dense (Dense)               multiple                  706885

=================================================================
Total params: 726277 (2.77 MB)
Trainable params: 726277 (2.77 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

As can be seen, the accuracy of both the test and the validation is not high, as it is a "toy" model and very simple, so it will not capture well certain characteristics of the images, resulting in this low accuracy.

2.2 Mid Complex Model

In [ ]:

```
mid_model = MidModel(num_classes=5)
optimizer = Adam(learning_rate=1e-3)
mid_model.compile(optimizer=optimizer, loss="sparse_categorical_crossentr
mid_model_history = train_model(
    mid_model,
    train_dataset,
    val_dataset,
    epochs=30,
    batch_size=BATCH_SIZE,
    verbose=1,
    path=model_dir,
    lr_patience=5,
    val_patience=10
).history
print(
    f"Final loss: training -> {mid_model_history['loss'][-1]:.2f}, valida
)
print(
    f"Final accuracy: training -> {mid_model_history['acc'][-1]:.2f}, val
)
```

```
Epoch 1/30
```

```
2024-03-07 11:57:58.546442: E tensorflow/core/grappler/optimizers/meta_opt
imizer.cc:961] layout failed: INVALID_ARGUMENT: Size of values 0 does not
match size of permutation 4 @ fanin shape inMidModel/dropout/dropout/Selec
tV2-2-TransposeNHWCToNCHW-LayoutOptimizer
```

```
45/45 [==============================] - 12s 135ms/step - loss: 1.3207 - a
cc: 0.4573 - val_loss: 0.9567 - val_acc: 0.6393 - lr: 0.0010
Epoch 2/30
45/45 [==============================] - 5s 86ms/step - loss: 0.8289 - acc
: 0.6846 - val_loss: 0.7033 - val_acc: 0.7338 - lr: 0.0010
Epoch 3/30
45/45 [==============================] - 5s 85ms/step - loss: 0.6622 - acc
: 0.7439 - val_loss: 0.6313 - val_acc: 0.7506 - lr: 0.0010
Epoch 4/30
45/45 [==============================] - 5s 86ms/step - loss: 0.5746 - acc
: 0.7725 - val_loss: 0.6024 - val_acc: 0.7719 - lr: 0.0010
Epoch 5/30
45/45 [==============================] - 5s 85ms/step - loss: 0.5285 - acc
: 0.7910 - val_loss: 0.5485 - val_acc: 0.7862 - lr: 0.0010
Epoch 6/30
45/45 [==============================] - 5s 86ms/step - loss: 0.4775 - acc
: 0.8122 - val_loss: 0.5855 - val_acc: 0.7803 - lr: 0.0010
Epoch 7/30
45/45 [==============================] - 5s 86ms/step - loss: 0.4587 - acc
: 0.8204 - val_loss: 0.5352 - val_acc: 0.8011 - lr: 0.0010
Epoch 8/30
45/45 [==============================] - 5s 86ms/step - loss: 0.4139 - acc
: 0.8376 - val_loss: 0.5116 - val_acc: 0.8046 - lr: 0.0010
Epoch 9/30
45/45 [==============================] - 5s 88ms/step - loss: 0.3732 - acc
: 0.8546 - val_loss: 0.4778 - val_acc: 0.8204 - lr: 0.0010
Epoch 10/30
45/45 [==============================] - 5s 85ms/step - loss: 0.3223 - acc
: 0.8759 - val_loss: 0.4984 - val_acc: 0.8144 - lr: 0.0010
Epoch 11/30
45/45 [==============================] - 5s 84ms/step - loss: 0.3063 - acc
: 0.8798 - val_loss: 0.5119 - val_acc: 0.8140 - lr: 0.0010
Epoch 12/30
45/45 [==============================] - 5s 85ms/step - loss: 0.2621 - acc
: 0.8983 - val_loss: 0.5139 - val_acc: 0.8238 - lr: 0.0010
Epoch 13/30
45/45 [==============================] - 5s 86ms/step - loss: 0.2232 - acc
: 0.9171 - val_loss: 0.5413 - val_acc: 0.8328 - lr: 0.0010
Epoch 14/30
45/45 [==============================] - 5s 86ms/step - loss: 0.1888 - acc
: 0.9258 - val_loss: 0.6093 - val_acc: 0.8204 - lr: 0.0010
Epoch 15/30
45/45 [==============================] - 5s 85ms/step - loss: 0.1175 - acc
: 0.9575 - val_loss: 0.5355 - val_acc: 0.8496 - lr: 2.0000e-04
Epoch 16/30
45/45 [==============================] - 5s 85ms/step - loss: 0.0861 - acc
: 0.9695 - val_loss: 0.5710 - val_acc: 0.8481 - lr: 2.0000e-04
Epoch 17/30
45/45 [==============================] - 5s 85ms/step - loss: 0.0767 - acc
: 0.9740 - val_loss: 0.6264 - val_acc: 0.8456 - lr: 2.0000e-04
Epoch 18/30
45/45 [==============================] - 5s 87ms/step - loss: 0.0663 - acc
: 0.9779 - val_loss: 0.6477 - val_acc: 0.8481 - lr: 2.0000e-04
Epoch 19/30
45/45 [==============================] - 5s 86ms/step - loss: 0.0583 - acc
: 0.9829 - val_loss: 0.6639 - val_acc: 0.8496 - lr: 2.0000e-04
Final loss: training -> 0.06, validation -> 0.66
Final accuracy: training -> 0.98, validation -> 0.85
```

In [ ]:

```
plot_history(mid_model_history, ['loss', 'acc'], name='Mid Complex Model'
```

In [ ]:
```
test_loss, test_accuracy = mid_model.evaluate(test_dataset, batch_size=BA
print(
    f"Model evaluated: Test Loss-> {test_loss}, Test Accuracy -> {test_ac
)
model_accuracies.append(('MidModel', test_accuracy))
mid_model.summary()
```

```
6/6 [==============================] - 1s 99ms/step - loss: 0.5026 - acc:
0.8764
Model evaluated: Test Loss-> 0.5025559067726135, Test Accuracy -> 87.64%
Model: "MidModel"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     multiple                  0

 conv2d_2 (Conv2D)           multiple                  896

 conv2d_3 (Conv2D)           multiple                  9248

 max_pooling2d_2 (MaxPoolin  multiple                  0
 g2D)

 dropout (Dropout)           multiple                  0

 conv2d_4 (Conv2D)           multiple                  18496

 conv2d_5 (Conv2D)           multiple                  36928

 conv2d_6 (Conv2D)           multiple                  73856

 conv2d_7 (Conv2D)           multiple                  147584

 flatten_1 (Flatten)         multiple                  0

 dense_1 (Dense)             multiple                  1327232

 dense_2 (Dense)             multiple                  645

=================================================================
Total params: 1614885 (6.16 MB)
Trainable params: 1614885 (6.16 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In the medium complexity model, it can be seen that it starts to overfit, although not to a great extent, from the middle of its epochs. This model has dropout, which means that this overfitting is reduced both in time and magnitude, benefiting the model, although with more complex models and other more sophisticated techniques its performance can be improved.

2.3 Model with residual connections

In [ ]:

```
resnet = CustomResNet(num_classes=5)
optimizer = Adam(learning_rate=1e-3)
resnet.compile(optimizer=optimizer, loss="sparse_categorical_crossentropy
resnet_history = train_model(
    resnet,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
    lr_patience=5,
```

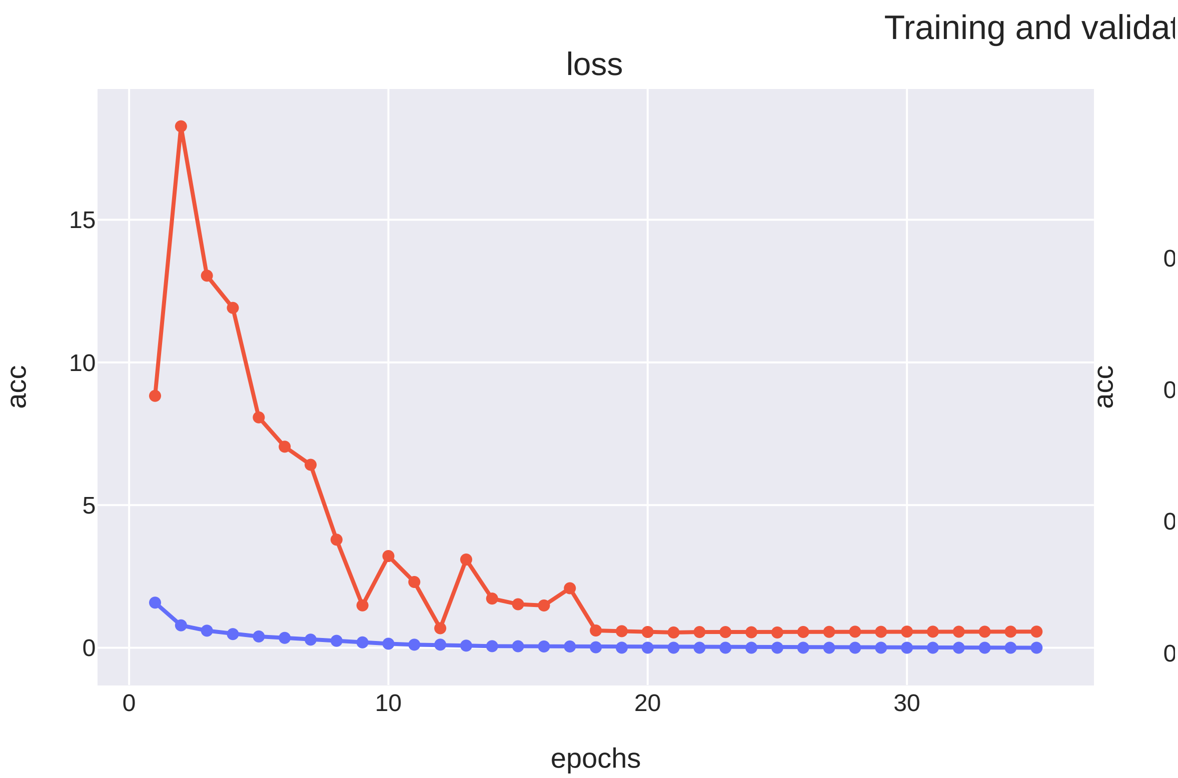
```
    val_patience=10,
).history
print(
    f"Final loss: training -> {resnet_history['loss'][-1]:.2f}, validatio
)
print(
    f"Final accuracy: training -> {resnet_history['acc'][-1]:.2f}, valida
)
```

```
Epoch 1/50
```

```
45/45 [==============================] - 14s 162ms/step - loss: 1.5817 - a
cc: 0.4257 - val_loss: 8.8287 - val_acc: 0.2088 - lr: 0.0010
Epoch 2/50
45/45 [==============================] - 5s 88ms/step - loss: 0.7860 - acc
: 0.6844 - val_loss: 18.2752 - val_acc: 0.2088 - lr: 0.0010
Epoch 3/50
45/45 [==============================] - 5s 87ms/step - loss: 0.5990 - acc
: 0.7636 - val_loss: 13.0407 - val_acc: 0.2088 - lr: 0.0010
Epoch 4/50
45/45 [==============================] - 5s 87ms/step - loss: 0.4790 - acc
: 0.8108 - val_loss: 11.9123 - val_acc: 0.2078 - lr: 0.0010
Epoch 5/50
45/45 [==============================] - 5s 95ms/step - loss: 0.3959 - acc
: 0.8446 - val_loss: 8.0744 - val_acc: 0.2345 - lr: 0.0010
Epoch 6/50
45/45 [==============================] - 5s 93ms/step - loss: 0.3466 - acc
: 0.8634 - val_loss: 7.0475 - val_acc: 0.2608 - lr: 0.0010
Epoch 7/50
45/45 [==============================] - 5s 94ms/step - loss: 0.2861 - acc
: 0.8895 - val_loss: 6.4130 - val_acc: 0.2217 - lr: 0.0010
Epoch 8/50
45/45 [==============================] - 5s 94ms/step - loss: 0.2437 - acc
: 0.9037 - val_loss: 3.7884 - val_acc: 0.3360 - lr: 0.0010
Epoch 9/50
45/45 [==============================] - 5s 95ms/step - loss: 0.1887 - acc
: 0.9285 - val_loss: 1.4852 - val_acc: 0.5804 - lr: 0.0010
Epoch 10/50
45/45 [==============================] - 5s 90ms/step - loss: 0.1418 - acc
: 0.9476 - val_loss: 3.2144 - val_acc: 0.4043 - lr: 0.0010
Epoch 11/50
45/45 [==============================] - 5s 86ms/step - loss: 0.1086 - acc
: 0.9584 - val_loss: 2.3050 - val_acc: 0.4790 - lr: 0.0010
Epoch 12/50
45/45 [==============================] - 5s 96ms/step - loss: 0.1096 - acc
: 0.9602 - val_loss: 0.6837 - val_acc: 0.7956 - lr: 0.0010
Epoch 13/50
45/45 [==============================] - 5s 88ms/step - loss: 0.0752 - acc
: 0.9728 - val_loss: 3.0924 - val_acc: 0.4923 - lr: 0.0010
Epoch 14/50
45/45 [==============================] - 5s 89ms/step - loss: 0.0557 - acc
: 0.9811 - val_loss: 1.7241 - val_acc: 0.6724 - lr: 0.0010
Epoch 15/50
45/45 [==============================] - 5s 89ms/step - loss: 0.0534 - acc
: 0.9814 - val_loss: 1.5253 - val_acc: 0.6685 - lr: 0.0010
Epoch 16/50
45/45 [==============================] - 5s 87ms/step - loss: 0.0455 - acc
: 0.9831 - val_loss: 1.4823 - val_acc: 0.7209 - lr: 0.0010
Epoch 17/50
45/45 [==============================] - 5s 86ms/step - loss: 0.0452 - acc
: 0.9844 - val_loss: 2.0859 - val_acc: 0.6091 - lr: 0.0010
Epoch 18/50
45/45 [==============================] - 5s 94ms/step - loss: 0.0164 - acc
: 0.9949 - val_loss: 0.6046 - val_acc: 0.8615 - lr: 2.0000e-04
Epoch 19/50
45/45 [==============================] - 5s 91ms/step - loss: 0.0031 - acc
: 0.9996 - val_loss: 0.5810 - val_acc: 0.8659 - lr: 2.0000e-04
Epoch 20/50
45/45 [==============================] - 5s 93ms/step - loss: 0.0018 - acc
: 0.9998 - val_loss: 0.5525 - val_acc: 0.8728 - lr: 2.0000e-04
Epoch 21/50
```

```
45/45 [==============================] - 5s 94ms/step - loss: 0.0012 - acc
: 0.9998 - val_loss: 0.5361 - val_acc: 0.8778 - lr: 2.0000e-04
Epoch 22/50
45/45 [==============================] - 5s 86ms/step - loss: 0.0012 - acc
: 0.9998 - val_loss: 0.5511 - val_acc: 0.8758 - lr: 2.0000e-04
Epoch 23/50
45/45 [==============================] - 5s 88ms/step - loss: 0.0011 - acc
: 0.9997 - val_loss: 0.5506 - val_acc: 0.8783 - lr: 2.0000e-04
Epoch 24/50
45/45 [==============================] - 5s 88ms/step - loss: 9.8771e-04 -
acc: 0.9997 - val_loss: 0.5432 - val_acc: 0.8793 - lr: 2.0000e-04
Epoch 25/50
45/45 [==============================] - 5s 94ms/step - loss: 8.4224e-04 -
acc: 0.9998 - val_loss: 0.5350 - val_acc: 0.8808 - lr: 2.0000e-04
Epoch 26/50
45/45 [==============================] - 5s 88ms/step - loss: 9.8657e-04 -
acc: 0.9997 - val_loss: 0.5533 - val_acc: 0.8798 - lr: 2.0000e-04
Epoch 27/50
45/45 [==============================] - 5s 85ms/step - loss: 7.0752e-04 -
acc: 0.9998 - val_loss: 0.5574 - val_acc: 0.8812 - lr: 2.0000e-04
Epoch 28/50
45/45 [==============================] - 5s 87ms/step - loss: 7.8093e-04 -
acc: 0.9998 - val_loss: 0.5640 - val_acc: 0.8788 - lr: 2.0000e-04
Epoch 29/50
45/45 [==============================] - 5s 89ms/step - loss: 0.0011 - acc
: 0.9997 - val_loss: 0.5594 - val_acc: 0.8808 - lr: 2.0000e-04
Epoch 30/50
45/45 [==============================] - 5s 87ms/step - loss: 5.3720e-04 -
acc: 0.9997 - val_loss: 0.5662 - val_acc: 0.8788 - lr: 2.0000e-04
Epoch 31/50
45/45 [==============================] - 5s 88ms/step - loss: 7.3354e-04 -
acc: 0.9998 - val_loss: 0.5647 - val_acc: 0.8803 - lr: 4.0000e-05
Epoch 32/50
45/45 [==============================] - 5s 85ms/step - loss: 4.7854e-04 -
acc: 0.9998 - val_loss: 0.5626 - val_acc: 0.8803 - lr: 4.0000e-05
Epoch 33/50
45/45 [==============================] - 5s 89ms/step - loss: 5.4136e-04 -
acc: 0.9999 - val_loss: 0.5634 - val_acc: 0.8808 - lr: 4.0000e-05
Epoch 34/50
45/45 [==============================] - 5s 88ms/step - loss: 6.9059e-04 -
acc: 0.9997 - val_loss: 0.5646 - val_acc: 0.8808 - lr: 4.0000e-05
Epoch 35/50
45/45 [==============================] - 5s 86ms/step - loss: 5.3650e-04 -
acc: 0.9998 - val_loss: 0.5660 - val_acc: 0.8808 - lr: 4.0000e-05
Final loss: training -> 0.00, validation -> 0.57
Final accuracy: training -> 1.00, validation -> 0.88
```

In [ ]:
```
plot_history(resnet_history, ['loss', 'acc'], name='Custom ResNet')
```



In [ ]:
```
test_loss, test_accuracy = resnet.evaluate(test_dataset, batch_size=BATCH_
print(
    f"Model evaluated: Test Loss-> {test_loss}, Test Accuracy -> {test_ac
)
model_accuracies.append(("ResNet", test_accuracy))
resnet.summary()
```

```
6/6 [==============================] - 1s 145ms/step - loss: 0.5762 - acc:
0.8891
Model evaluated: Test Loss-> 0.5761938095092773, Test Accuracy -> 88.91%
Model: "ResNet"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 rescaling_2 (Rescaling)     multiple                  0         
                                                                 
 zero_padding2d (ZeroPaddin  multiple                  0         
 g2D)                                                            
                                                                 
 conv2d_8 (Conv2D)           multiple                  9472      
                                                                 
 batch_normalization (Batch  multiple                  256       
 Normalization)                                                  
                                                                 
 max_pooling2d_3 (MaxPoolin  multiple                  0         
 g2D)                                                            
                                                                 
 identity_block (IdentityBl  multiple                  74368     
 ock)                                                            
                                                                 
 identity_block_1 (Identity  multiple                  74368     
 Block)                                                          
                                                                 
 convolution_block (Convolu  multiple                  230784    
 tionBlock)                                                      
                                                                 
 identity_block_2 (Identity  multiple                  296192    
 Block)                                                          
                                                                 
 convolution_block_1 (Convo  multiple                  920320    
 lutionBlock)                                                    
                                                                 
 identity_block_3 (Identity  multiple                  1182208   
 Block)                                                          
                                                                 
 convolution_block_2 (Convo  multiple                  3675648   
 lutionBlock)                                                    
                                                                 
 identity_block_4 (Identity  multiple                  4723712   
 Block)                                                          
                                                                 
 dense_3 (Dense)             multiple                  513000    
                                                                 
 dense_4 (Dense)             multiple                  5005      
                                                                 
=================================================================
Total params: 11705333 (44.65 MB)
Trainable params: 11697525 (44.62 MB)
Non-trainable params: 7808 (30.50 KB)
_________________________________________________________________
```

Among the more complex models we have decided on two customised implementations, a model with residual connections and a model based on the "inception" technique, with its corresponding blocks. This allows these models to be deeper without the problem of gradient fading and can better capture the characteristics of the data and result in better performance. We first tested a residual model with 4 blocks and filter size 64, which already gives a better accuracy than the previous ones. This model would be an "approximation" to a resnet14, although differing in certain aspects. It can be seen that the model does not overfit despite having a large number of epochs, which is a good indication.

2.4. Inception block

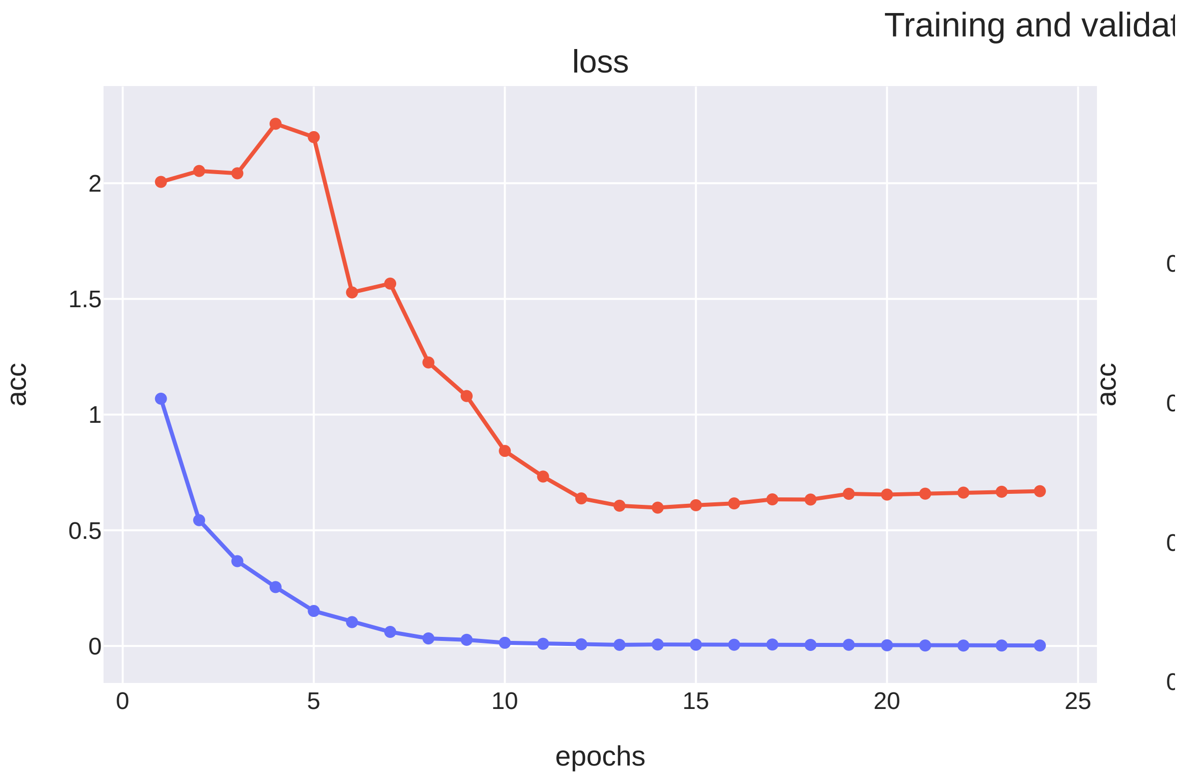
In [ ]:
```
inception = InceptionModel(num_classes=5, num_blocks=3, n_filters=[(32, 3
optimizer = Adam(learning_rate=1e-4)
inception.compile(optimizer=optimizer, loss="sparse_categorical_crossentr
inception_history = train_model(
    inception,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
    lr_patience=5,
    val_patience=10
).history
print(
    f"Final loss: training -> {inception_history['loss'][-1]:.2f}, valida
)
print(
    f"Final accuracy: training -> {inception_history['acc'][-1]:.2f}, val
)
```

```
Epoch 1/50
```

```
45/45 [==============================] - 106s 1s/step - loss: 1.0687 - acc
: 0.6346 - val_loss: 2.0058 - val_acc: 0.2519 - lr: 1.0000e-04
Epoch 2/50
45/45 [==============================] - 17s 361ms/step - loss: 0.5435 - a
cc: 0.7950 - val_loss: 2.0530 - val_acc: 0.2736 - lr: 1.0000e-04
Epoch 3/50
45/45 [==============================] - 17s 362ms/step - loss: 0.3665 - a
cc: 0.8637 - val_loss: 2.0429 - val_acc: 0.2810 - lr: 1.0000e-04
Epoch 4/50
45/45 [==============================] - 17s 361ms/step - loss: 0.2545 - a
cc: 0.9069 - val_loss: 2.2570 - val_acc: 0.2677 - lr: 1.0000e-04
Epoch 5/50
45/45 [==============================] - 17s 361ms/step - loss: 0.1513 - a
cc: 0.9494 - val_loss: 2.1995 - val_acc: 0.3597 - lr: 1.0000e-04
Epoch 6/50
45/45 [==============================] - 17s 368ms/step - loss: 0.1031 - a
cc: 0.9676 - val_loss: 1.5280 - val_acc: 0.5191 - lr: 1.0000e-04
Epoch 7/50
45/45 [==============================] - 17s 361ms/step - loss: 0.0608 - a
cc: 0.9837 - val_loss: 1.5663 - val_acc: 0.4696 - lr: 1.0000e-04
Epoch 8/50
45/45 [==============================] - 18s 368ms/step - loss: 0.0326 - a
cc: 0.9944 - val_loss: 1.2253 - val_acc: 0.5764 - lr: 1.0000e-04
Epoch 9/50
45/45 [==============================] - 17s 368ms/step - loss: 0.0266 - a
cc: 0.9955 - val_loss: 1.0802 - val_acc: 0.6363 - lr: 1.0000e-04
Epoch 10/50
45/45 [==============================] - 17s 368ms/step - loss: 0.0137 - a
cc: 0.9992 - val_loss: 0.8429 - val_acc: 0.7219 - lr: 1.0000e-04
Epoch 11/50
45/45 [==============================] - 17s 368ms/step - loss: 0.0094 - a
cc: 0.9997 - val_loss: 0.7322 - val_acc: 0.7694 - lr: 1.0000e-04
Epoch 12/50
45/45 [==============================] - 17s 368ms/step - loss: 0.0073 - a
cc: 0.9997 - val_loss: 0.6378 - val_acc: 0.8070 - lr: 1.0000e-04
Epoch 13/50
45/45 [==============================] - 18s 368ms/step - loss: 0.0048 - a
cc: 0.9997 - val_loss: 0.6061 - val_acc: 0.8135 - lr: 1.0000e-04
Epoch 14/50
45/45 [==============================] - 17s 368ms/step - loss: 0.0063 - a
cc: 0.9997 - val_loss: 0.5978 - val_acc: 0.8258 - lr: 1.0000e-04
Epoch 15/50
45/45 [==============================] - 17s 361ms/step - loss: 0.0054 - a
cc: 0.9997 - val_loss: 0.6083 - val_acc: 0.8318 - lr: 1.0000e-04
Epoch 16/50
45/45 [==============================] - 17s 360ms/step - loss: 0.0052 - a
cc: 0.9997 - val_loss: 0.6162 - val_acc: 0.8357 - lr: 1.0000e-04
Epoch 17/50
45/45 [==============================] - 17s 361ms/step - loss: 0.0062 - a
cc: 0.9997 - val_loss: 0.6337 - val_acc: 0.8382 - lr: 1.0000e-04
Epoch 18/50
45/45 [==============================] - 17s 361ms/step - loss: 0.0046 - a
cc: 0.9997 - val_loss: 0.6328 - val_acc: 0.8357 - lr: 1.0000e-04
Epoch 19/50
45/45 [==============================] - 17s 361ms/step - loss: 0.0050 - a
cc: 0.9997 - val_loss: 0.6576 - val_acc: 0.8323 - lr: 1.0000e-04
Epoch 20/50
45/45 [==============================] - 17s 361ms/step - loss: 0.0028 - a
cc: 0.9998 - val_loss: 0.6543 - val_acc: 0.8382 - lr: 2.0000e-05
Epoch 21/50
```

```
45/45 [==============================] - 17s 361ms/step - loss: 0.0021 - a
cc: 0.9997 - val_loss: 0.6582 - val_acc: 0.8382 - lr: 2.0000e-05
Epoch 22/50
45/45 [==============================] - 17s 361ms/step - loss: 0.0018 - a
cc: 0.9997 - val_loss: 0.6628 - val_acc: 0.8382 - lr: 2.0000e-05
Epoch 23/50
45/45 [==============================] - 17s 361ms/step - loss: 0.0019 - a
cc: 0.9997 - val_loss: 0.6665 - val_acc: 0.8422 - lr: 2.0000e-05
Epoch 24/50
45/45 [==============================] - 17s 361ms/step - loss: 0.0020 - a
cc: 0.9997 - val_loss: 0.6690 - val_acc: 0.8402 - lr: 2.0000e-05
Final loss: training -> 0.00, validation -> 0.67
Final accuracy: training -> 1.00, validation -> 0.84
```

In [ ]:
```
plot_history(inception_history, ['loss', 'acc'], name='Inception Model')
```



In [ ]:
```
test_loss, test_accuracy = inception.evaluate(test_dataset, batch_size=BA
print(
    f"Model evaluated: Test Loss -> {test_loss}, Test Accuracy -> {test_a
)
model_accuracies.append(("Inception", test_accuracy))
inception.summary()
```

```
6/6 [==============================] - 7s 1s/step - loss: 0.5857 - acc: 0.
8484
Model evaluated: Test Loss -> 0.5856717228889465, Test Accuracy -> 84.84%
Model: "InceptionModel"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_3 (Rescaling)     multiple                  0

 inception (Inception)       multiple                  83872

 inception_1 (Inception)     multiple                  95584

 inception_2 (Inception)     multiple                  95584

 conv2d_35 (Conv2D)          multiple                  602176

 conv2d_43 (Conv2D)          multiple                  602176

 conv2d_51 (Conv2D)          multiple                  602176

 batch_normalization_17 (Ba  multiple                  768
 tchNormalization)

 batch_normalization_18 (Ba  multiple                  768
 tchNormalization)

 batch_normalization_19 (Ba  multiple                  768
 tchNormalization)

 dense_5 (Dense)             multiple                  12545000

 dropout_1 (Dropout)         multiple                  0

 dense_6 (Dense)             multiple                  5005

=================================================================
Total params: 14633877 (55.82 MB)
Trainable params: 14632725 (55.82 MB)
Non-trainable params: 1152 (4.50 KB)
_________________________________________________________________
```

For the second deep model, an inception architecture has been chosen. It can be seen in the graphs how at the beginning it was difficult to start generalising, as reflected in the metrics of the validation set, but as the model learned, it also started to generalise, although its accuracy does not surpass that of the residual model. This model does not overfit either, but because of the above, it is somewhat distant from training performance.

2.5 What if we add Data Augmentation?

In [ ]:
```
from keras.models import Sequential
resnet_data_aug = Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.2),
    layers.GaussianNoise(0.01),
```

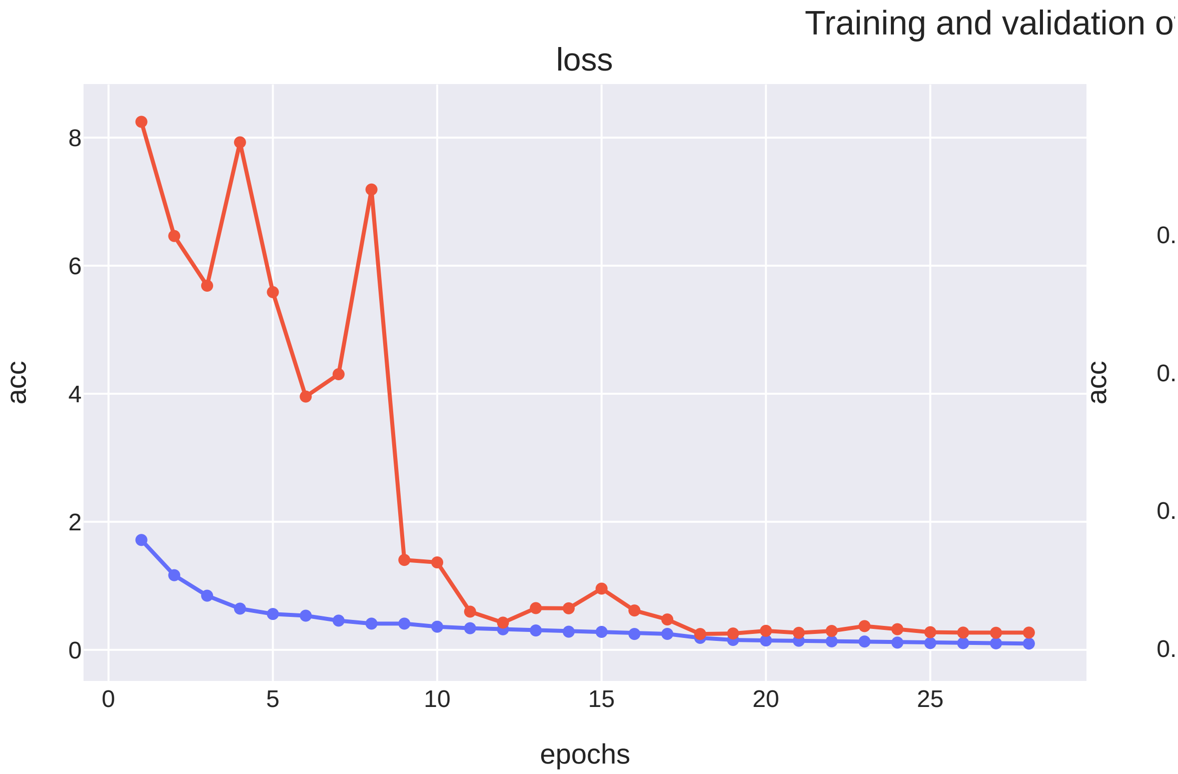
```
    CustomResNet(num_classes=5)
], name='ResNetAug')

resnet_data_aug.compile(optimizer="adam", loss="sparse_categorical_crosse
resnet_data_aug_history = train_model(
    resnet_data_aug,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
    lr_patience=5,
    val_patience=10,
).history

print(
    f"Final loss: training -> {resnet_data_aug_history['loss'][-1]:.2f},
)
print(
    f"Final accuracy: training -> {resnet_data_aug_history['acc'][-1]:.2f
)
```

```
Epoch 1/50
```

```
45/45 [==============================] - 9s 99ms/step - loss: 1.7157 - acc
: 0.3624 - val_loss: 8.2472 - val_acc: 0.2088 - lr: 0.0010
Epoch 2/50
45/45 [==============================] - 5s 92ms/step - loss: 1.1655 - acc
: 0.5241 - val_loss: 6.4635 - val_acc: 0.2068 - lr: 0.0010
Epoch 3/50
45/45 [==============================] - 5s 95ms/step - loss: 0.8457 - acc
: 0.6654 - val_loss: 5.6873 - val_acc: 0.2143 - lr: 0.0010
Epoch 4/50
45/45 [==============================] - 5s 90ms/step - loss: 0.6436 - acc
: 0.7465 - val_loss: 7.9266 - val_acc: 0.2815 - lr: 0.0010
Epoch 5/50
45/45 [==============================] - 5s 96ms/step - loss: 0.5598 - acc
: 0.7761 - val_loss: 5.5866 - val_acc: 0.2825 - lr: 0.0010
Epoch 6/50
45/45 [==============================] - 5s 92ms/step - loss: 0.5336 - acc
: 0.7929 - val_loss: 3.9558 - val_acc: 0.2578 - lr: 0.0010
Epoch 7/50
45/45 [==============================] - 5s 89ms/step - loss: 0.4562 - acc
: 0.8275 - val_loss: 4.3048 - val_acc: 0.2905 - lr: 0.0010
Epoch 8/50
45/45 [==============================] - 5s 88ms/step - loss: 0.4093 - acc
: 0.8400 - val_loss: 7.1897 - val_acc: 0.2380 - lr: 0.0010
Epoch 9/50
45/45 [==============================] - 5s 95ms/step - loss: 0.4094 - acc
: 0.8417 - val_loss: 1.4042 - val_acc: 0.5923 - lr: 0.0010
Epoch 10/50
45/45 [==============================] - 5s 94ms/step - loss: 0.3613 - acc
: 0.8601 - val_loss: 1.3657 - val_acc: 0.5710 - lr: 0.0010
Epoch 11/50
45/45 [==============================] - 5s 96ms/step - loss: 0.3377 - acc
: 0.8718 - val_loss: 0.5966 - val_acc: 0.7768 - lr: 0.0010
Epoch 12/50
45/45 [==============================] - 5s 95ms/step - loss: 0.3205 - acc
: 0.8768 - val_loss: 0.4250 - val_acc: 0.8427 - lr: 0.0010
Epoch 13/50
45/45 [==============================] - 5s 89ms/step - loss: 0.3024 - acc
: 0.8861 - val_loss: 0.6516 - val_acc: 0.7679 - lr: 0.0010
Epoch 14/50
45/45 [==============================] - 5s 89ms/step - loss: 0.2835 - acc
: 0.8902 - val_loss: 0.6484 - val_acc: 0.7640 - lr: 0.0010
Epoch 15/50
45/45 [==============================] - 5s 90ms/step - loss: 0.2803 - acc
: 0.8905 - val_loss: 0.9562 - val_acc: 0.7125 - lr: 0.0010
Epoch 16/50
45/45 [==============================] - 5s 87ms/step - loss: 0.2480 - acc
: 0.9068 - val_loss: 0.6148 - val_acc: 0.7937 - lr: 0.0010
Epoch 17/50
45/45 [==============================] - 5s 90ms/step - loss: 0.2491 - acc
: 0.9051 - val_loss: 0.4744 - val_acc: 0.8238 - lr: 0.0010
Epoch 18/50
45/45 [==============================] - 5s 95ms/step - loss: 0.1869 - acc
: 0.9308 - val_loss: 0.2474 - val_acc: 0.9075 - lr: 2.0000e-04
Epoch 19/50
45/45 [==============================] - 5s 89ms/step - loss: 0.1544 - acc
: 0.9431 - val_loss: 0.2554 - val_acc: 0.9060 - lr: 2.0000e-04
Epoch 20/50
45/45 [==============================] - 5s 87ms/step - loss: 0.1467 - acc
: 0.9467 - val_loss: 0.2968 - val_acc: 0.8882 - lr: 2.0000e-04
Epoch 21/50
```

```
45/45 [==============================] - 5s 86ms/step - loss: 0.1417 - acc
: 0.9488 - val_loss: 0.2656 - val_acc: 0.9015 - lr: 2.0000e-04
Epoch 22/50
45/45 [==============================] - 5s 89ms/step - loss: 0.1326 - acc
: 0.9527 - val_loss: 0.2950 - val_acc: 0.8902 - lr: 2.0000e-04
Epoch 23/50
45/45 [==============================] - 5s 91ms/step - loss: 0.1300 - acc
: 0.9514 - val_loss: 0.3706 - val_acc: 0.8758 - lr: 2.0000e-04
Epoch 24/50
45/45 [==============================] - 5s 89ms/step - loss: 0.1131 - acc
: 0.9572 - val_loss: 0.3223 - val_acc: 0.8906 - lr: 4.0000e-05
Epoch 25/50
45/45 [==============================] - 5s 91ms/step - loss: 0.1058 - acc
: 0.9612 - val_loss: 0.2754 - val_acc: 0.9075 - lr: 4.0000e-05
Epoch 26/50
45/45 [==============================] - 5s 89ms/step - loss: 0.1047 - acc
: 0.9625 - val_loss: 0.2694 - val_acc: 0.9080 - lr: 4.0000e-05
Epoch 27/50
45/45 [==============================] - 5s 90ms/step - loss: 0.1015 - acc
: 0.9635 - val_loss: 0.2663 - val_acc: 0.9070 - lr: 4.0000e-05
Epoch 28/50
45/45 [==============================] - 5s 87ms/step - loss: 0.0978 - acc
: 0.9640 - val_loss: 0.2690 - val_acc: 0.9095 - lr: 4.0000e-05
Final loss: training -> 0.10, validation -> 0.27
Final accuracy: training -> 0.96, validation -> 0.91
```

In [ ]:
```
plot_history(resnet_data_aug_history, ['loss', 'acc'], name='Augmented Re
```



In [ ]:
```
test_loss, test_accuracy = resnet_data_aug.evaluate(test_dataset, batch_s
print(
    f"Model evaluated: Test Loss-> {test_loss}, Test Accuracy -> {test_ac
)
model_accuracies.append(("AugmentedResNet", test_accuracy))
resnet_data_aug.summary()
```

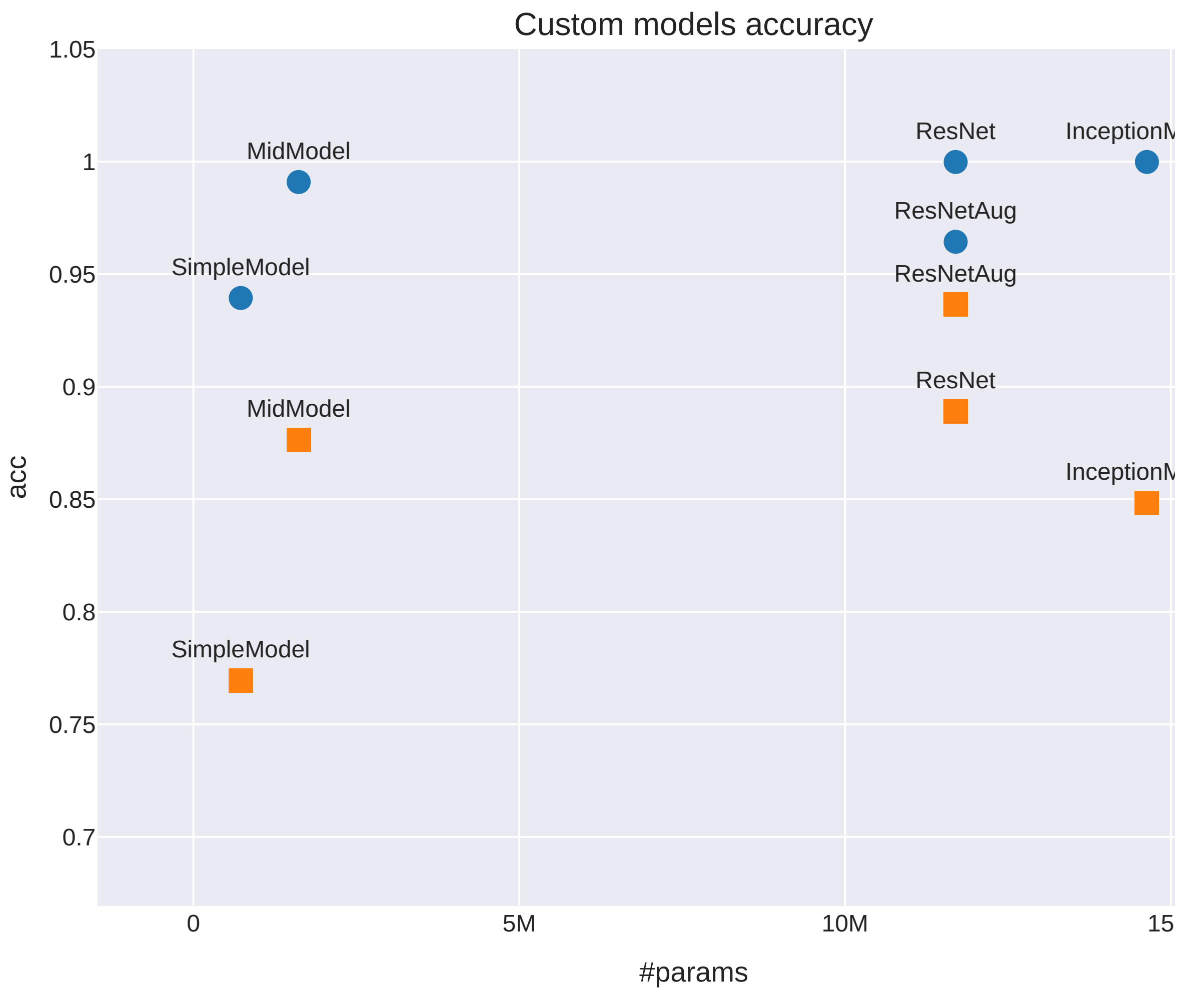
```
6/6 [==============================] - 0s 21ms/step - loss: 0.2016 - acc:
0.9365
Model evaluated: Test Loss-> 0.20161490142345428, Test Accuracy -> 93.65%
Model: "ResNetAug"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 random_flip (RandomFlip)    (None, 100, 100, 3)       0

 random_rotation (RandomRot  (None, 100, 100, 3)       0
 ation)

 random_zoom (RandomZoom)    (None, 100, 100, 3)       0

 gaussian_noise (GaussianNo  (None, 100, 100, 3)       0
 ise)

 ResNet (CustomResNet)       (None, 5)                 11705333

=================================================================
Total params: 11705333 (44.65 MB)
Trainable params: 11697525 (44.62 MB)
Non-trainable params: 7808 (30.50 KB)
_________________________________________________________________
```

To enhance the model's generalisation and prevent it from overfitting to the training set, we decided to use data augmentation. We selected the residual model, which was the best model available, and added four stages of data augmentation: horizontal flip, rotation, zoom, and Gaussian noise. The results demonstrate that this technique has significantly improved the model's performance, achieving over 90% accuracy in the test set, which is impressive for a non-pretrained model. Additionally, the model appears to be free of overfitting, as evidenced by the consistent loss of training and validation across epochs.

2.6. Custom models comparison

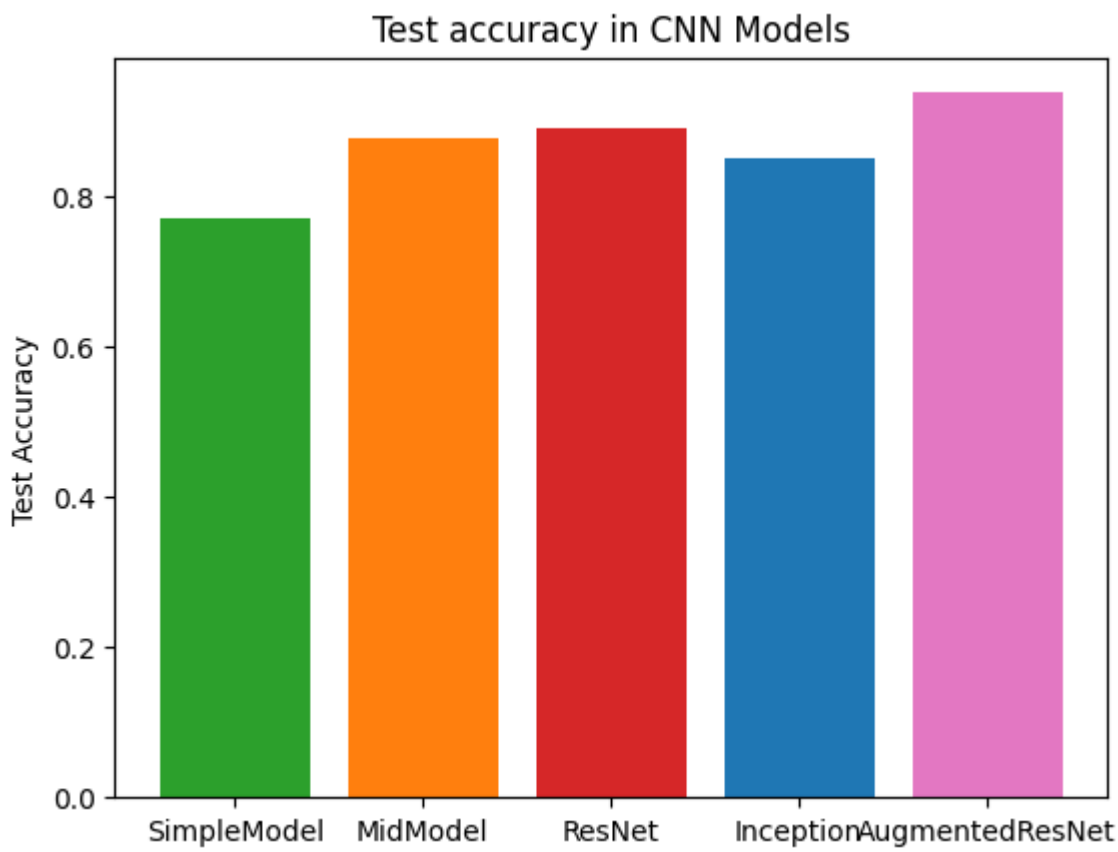
In the next output we have displayed the accuracy performance of each custom model in the train and test set to analyze the overfitting behavior.

In [ ]:
```
models = [simple_model, mid_model, inception, resnet, resnet_data_aug]
comparison(models, (train_dataset, test_dataset))
```



The left figure shows each model performance in the train (blue) and test (orange) sets. We see that for all models the performance is considerably better in the train set (99%) than in the evaluation samples (75-85%), proving the network is learning the train set characteristics instead of generalizing image information for the given task. The right figure shows the disadvantage of using larger networks with more complex connections (Inception), inducing a higher difference between the train and test set. In the other hand, we can see that a simple model like the first that was trained, without any type of regularization technique, is clearly overfitted, shows a great difference meaning that simpler models tends to learn the training set instead of learning to generalize. Additionally, we see the impact of regularizing the network via data augmentation (ResNetAug). The result is the smaller difference between the train and test set.

In [ ]:
```
fig, ax = plt.subplots()
models = [name for name, _ in model_accuracies]
accuracies = [accuracy for _, accuracy in model_accuracies]
bar_colors = ["tab:green", "tab:orange", "tab:red", "tab:blue", 'tab:pink
ax.bar(models, accuracies, label=models, color=bar_colors)

ax.set_ylabel("Test Accuracy")
ax.set_title("Test accuracy in CNN Models")
plt.show()
```



Exercise 3. Pretrained models

For this exercise we conducted experiments with three pretrained models available in the Keras Applications website pretrained with ImageNet dataset: ResNet-50 (He et al., 2016), MobileNet-V2 (Sandler et al.,, 2018) and EfficientNet-B0 (Tan & Le, 2019). Our first approach uses transfer learning, using the pretrained models freezed to use that previous knowledge and only train our classifier. The second round is fine-tuning, where we train the model with the feature-extractor freezed, to train the top (classifier) and then do a little round of epochs with very low learning rate with the pretrained model unfreezed to adapt its weights to our dataset. The last approaches explores defreezing only the last 10 layers of each architecture to test if maintaining a large ratio of trainable weights does not impact in the overal performance.

3.1. Transfer-learning

ResNet-50 V2

In [ ]:
```
resnet_t = PretrainedModel(num_classes=5, img_size=IMG_SIZE, pretrained='
optimizer = Adam(learning_rate=1e-4)
resnet_t.compile(optimizer=optimizer, loss="sparse_categorical_crossentro
resnet_t_history = train_model(
    resnet_t,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
```

```
    lr_patience=5,
    val_patience=10
).history
print(
    f"Final loss: training -> {resnet_t_history['loss'][-1]:.2f}, validat
)
print(
    f"Final accuracy: training -> {resnet_t_history['acc'][-1]:.2f}, vali
)
test_loss, test_accuracy = resnet_t.evaluate(test_dataset, batch_size=BAT
print(
    f"Model evaluated: Test Loss-> {test_loss}, Test Accuracy -> {test_ac
)
resnet_t.summary()
```

```
Epoch 1/50
45/45 [==============================] - 9s 136ms/step - loss: 0.8935 - ac
c: 0.8552 - val_loss: 0.2917 - val_acc: 0.9095 - lr: 1.0000e-04
Epoch 2/50
45/45 [==============================] - 6s 109ms/step - loss: 0.0984 - ac
c: 0.9701 - val_loss: 0.2475 - val_acc: 0.9184 - lr: 1.0000e-04
Epoch 3/50
45/45 [==============================] - 5s 96ms/step - loss: 0.0243 - acc
: 0.9960 - val_loss: 0.2590 - val_acc: 0.9198 - lr: 1.0000e-04
Epoch 4/50
45/45 [==============================] - 5s 93ms/step - loss: 0.0104 - acc
: 0.9994 - val_loss: 0.2704 - val_acc: 0.9198 - lr: 1.0000e-04
Epoch 5/50
45/45 [==============================] - 5s 93ms/step - loss: 0.0064 - acc
: 0.9997 - val_loss: 0.2796 - val_acc: 0.9213 - lr: 1.0000e-04
Epoch 6/50
45/45 [==============================] - 5s 92ms/step - loss: 0.0043 - acc
: 0.9998 - val_loss: 0.2883 - val_acc: 0.9198 - lr: 1.0000e-04
Epoch 7/50
45/45 [==============================] - 5s 92ms/step - loss: 0.0036 - acc
: 0.9998 - val_loss: 0.2921 - val_acc: 0.9218 - lr: 1.0000e-04
Epoch 8/50
45/45 [==============================] - 5s 89ms/step - loss: 0.0029 - acc
: 0.9998 - val_loss: 0.2942 - val_acc: 0.9198 - lr: 2.0000e-05
Epoch 9/50
45/45 [==============================] - 5s 91ms/step - loss: 0.0026 - acc
: 0.9998 - val_loss: 0.2956 - val_acc: 0.9189 - lr: 2.0000e-05
Epoch 10/50
45/45 [==============================] - 5s 91ms/step - loss: 0.0025 - acc
: 0.9997 - val_loss: 0.2968 - val_acc: 0.9193 - lr: 2.0000e-05
Epoch 11/50
45/45 [==============================] - 5s 92ms/step - loss: 0.0024 - acc
: 0.9997 - val_loss: 0.2976 - val_acc: 0.9193 - lr: 2.0000e-05
Epoch 12/50
45/45 [==============================] - 5s 90ms/step - loss: 0.0023 - acc
: 0.9997 - val_loss: 0.2988 - val_acc: 0.9203 - lr: 2.0000e-05
Final loss: training -> 0.00, validation -> 0.30
Final accuracy: training -> 1.00, validation -> 0.92
6/6 [==============================] - 1s 142ms/step - loss: 0.2515 - acc:
0.9446
Model evaluated: Test Loss-> 0.25145092606544495, Test Accuracy -> 94.46%
Model: "ResNet-t"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_5 (Rescaling)     multiple                  0

 resnet50v2 (Functional)     (None, 4, 4, 2048)        23564800

 flatten_2 (Flatten)         multiple                  0

 dense_9 (Dense)             multiple                  32769000

 dense_10 (Dense)            multiple                  5005

=================================================================
Total params: 56338805 (214.92 MB)
Trainable params: 32774005 (125.02 MB)
Non-trainable params: 23564800 (89.89 MB)
_________________________________________________________________
```

In [ ]:
```
plot_history(resnet_t_history, ['loss', 'acc'], name=f'{resnet_t.name} Mo
```

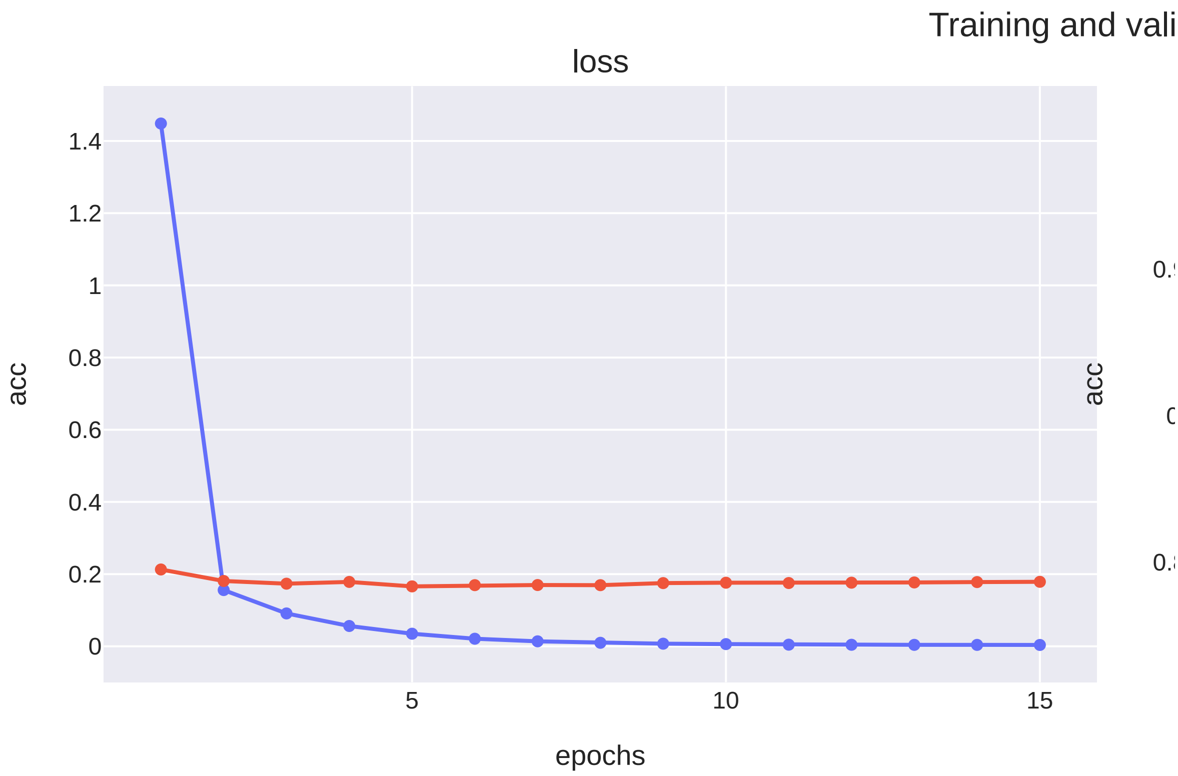
MobileNet V2

In [ ]:
```
mobile_t = PretrainedModel(num_classes=5, img_size=IMG_SIZE, pretrained='
optimizer = Adam(learning_rate=1e-4)
mobile_t.compile(optimizer=optimizer, loss="sparse_categorical_crossentro
mobile_t_history = train_model(
    mobile_t,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
    lr_patience=5,
    val_patience=10
).history
print(
    f"Final loss: training -> {mobile_t_history['loss'][-1]:.2f}, validat
)
print(
    f"Final accuracy: training -> {mobile_t_history['acc'][-1]:.2f}, vali
)
test_loss, test_accuracy = mobile_t.evaluate(test_dataset, batch_size=BAT
print(
    f"Model evaluated: Test Loss-> {test_loss}, Test Accuracy -> {test_ac
)
mobile_t.summary()
```

```
Epoch 1/50
45/45 [==============================] - 7s 106ms/step - loss: 1.4484 - ac
c: 0.8216 - val_loss: 0.2129 - val_acc: 0.9139 - lr: 1.0000e-04
Epoch 2/50
45/45 [==============================] - 5s 96ms/step - loss: 0.1558 - acc
: 0.9416 - val_loss: 0.1809 - val_acc: 0.9233 - lr: 1.0000e-04
Epoch 3/50
45/45 [==============================] - 5s 97ms/step - loss: 0.0911 - acc
: 0.9706 - val_loss: 0.1735 - val_acc: 0.9317 - lr: 1.0000e-04
Epoch 4/50
45/45 [==============================] - 5s 89ms/step - loss: 0.0563 - acc
: 0.9862 - val_loss: 0.1784 - val_acc: 0.9352 - lr: 1.0000e-04
Epoch 5/50
45/45 [==============================] - 5s 98ms/step - loss: 0.0349 - acc
: 0.9954 - val_loss: 0.1660 - val_acc: 0.9396 - lr: 1.0000e-04
Epoch 6/50
45/45 [==============================] - 5s 87ms/step - loss: 0.0210 - acc
: 0.9984 - val_loss: 0.1695 - val_acc: 0.9436 - lr: 1.0000e-04
Epoch 7/50
45/45 [==============================] - 5s 87ms/step - loss: 0.0138 - acc
: 0.9992 - val_loss: 0.1698 - val_acc: 0.9446 - lr: 1.0000e-04
Epoch 8/50
45/45 [==============================] - 5s 85ms/step - loss: 0.0098 - acc
: 0.9997 - val_loss: 0.1694 - val_acc: 0.9431 - lr: 1.0000e-04
Epoch 9/50
45/45 [==============================] - 5s 86ms/step - loss: 0.0073 - acc
: 0.9997 - val_loss: 0.1751 - val_acc: 0.9426 - lr: 1.0000e-04
Epoch 10/50
45/45 [==============================] - 5s 87ms/step - loss: 0.0061 - acc
: 0.9997 - val_loss: 0.1762 - val_acc: 0.9416 - lr: 1.0000e-04
Epoch 11/50
45/45 [==============================] - 5s 87ms/step - loss: 0.0045 - acc
: 0.9998 - val_loss: 0.1753 - val_acc: 0.9426 - lr: 2.0000e-05
Epoch 12/50
45/45 [==============================] - 5s 86ms/step - loss: 0.0043 - acc
: 0.9998 - val_loss: 0.1763 - val_acc: 0.9431 - lr: 2.0000e-05
Epoch 13/50
45/45 [==============================] - 5s 86ms/step - loss: 0.0040 - acc
: 0.9998 - val_loss: 0.1769 - val_acc: 0.9441 - lr: 2.0000e-05
Epoch 14/50
45/45 [==============================] - 5s 86ms/step - loss: 0.0039 - acc
: 0.9997 - val_loss: 0.1780 - val_acc: 0.9426 - lr: 2.0000e-05
Epoch 15/50
45/45 [==============================] - 5s 88ms/step - loss: 0.0038 - acc
: 0.9997 - val_loss: 0.1787 - val_acc: 0.9426 - lr: 2.0000e-05
Final loss: training -> 0.00, validation -> 0.18
Final accuracy: training -> 1.00, validation -> 0.94
6/6 [==============================] - 1s 46ms/step - loss: 0.1589 - acc:
0.9519
Model evaluated: Test Loss-> 0.1589222103357315, Test Accuracy -> 95.19%
Model: "Mobile-t"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 rescaling_6 (Rescaling)     multiple                  0         
                                                                 
 mobilenetv2_1.00_224 (Func  (None, 4, 4, 1280)        2257984   
 tional)                                                         
                                                                 
 flatten_3 (Flatten)         multiple                  0         
```

```
 dense_11 (Dense)            multiple                  20481000

 dense_12 (Dense)            multiple                  5005

=================================================================
Total params: 22743989 (86.76 MB)
Trainable params: 20486005 (78.15 MB)
Non-trainable params: 2257984 (8.61 MB)
_________________________________________________________________
```

In [ ]:
```
plot_history(mobile_t_history, ['loss', 'acc'], name=f'{mobile_t.name} Mo
```



Xception

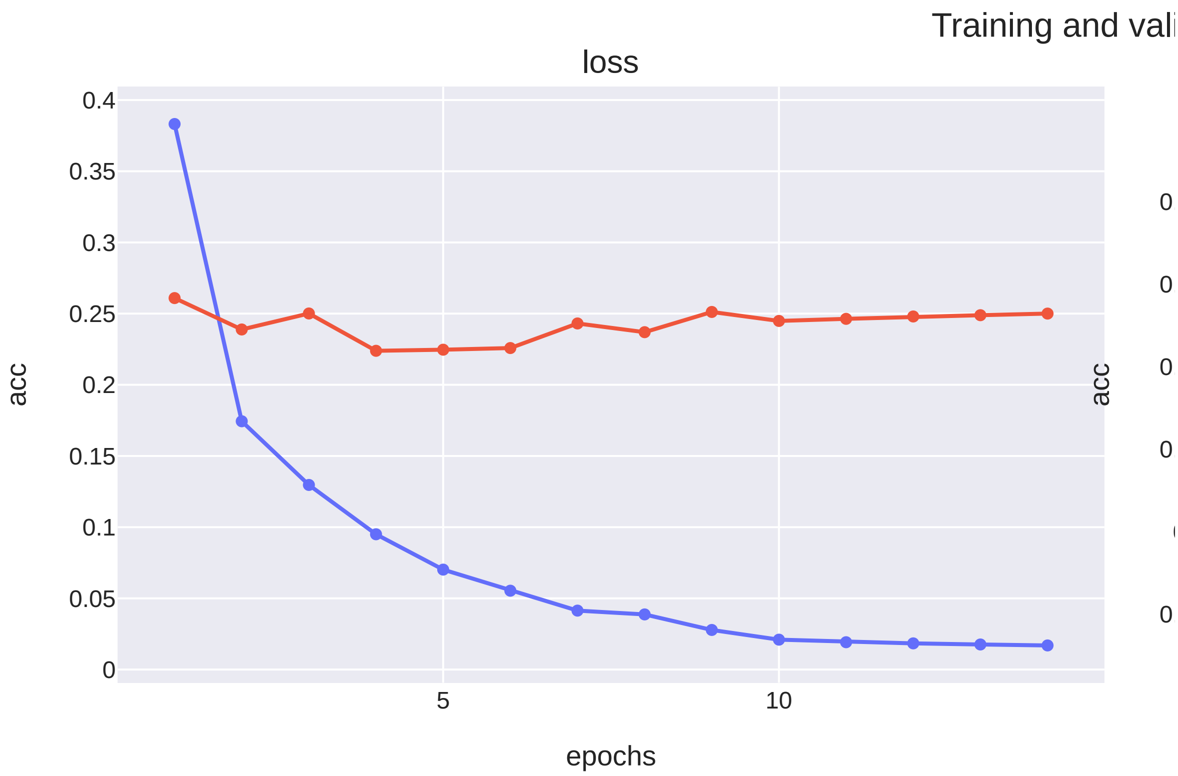
In [ ]:
```
xception_t = PretrainedModel(num_classes=5, img_size=IMG_SIZE, pretrained
optimizer = Adam(learning_rate=1e-4)
xception_t.compile(optimizer=optimizer, loss="sparse_categorical_crossent
xception_t_history = train_model(
    xception_t,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
    lr_patience=5,
    val_patience=10
).history
print(
    f"Final loss: training -> {xception_t_history['loss'][-1]:.2f}, valid
)
print(
    f"Final accuracy: training -> {xception_t_history['acc'][-1]:.2f}, va
)
test_loss, test_accuracy = xception_t.evaluate(test_dataset, batch_size=B
```

```
print(
    f"Model evaluated: Test Loss-> {test_loss}, Test Accuracy -> {test_ac
)
xception_t.summary()
```

```
Epoch 1/50
45/45 [==============================] - 8s 120ms/step - loss: 0.3832 - ac
c: 0.8724 - val_loss: 0.2609 - val_acc: 0.8996 - lr: 1.0000e-04
Epoch 2/50
45/45 [==============================] - 6s 104ms/step - loss: 0.1744 - ac
c: 0.9361 - val_loss: 0.2389 - val_acc: 0.9134 - lr: 1.0000e-04
Epoch 3/50
45/45 [==============================] - 5s 92ms/step - loss: 0.1297 - acc
: 0.9548 - val_loss: 0.2501 - val_acc: 0.9095 - lr: 1.0000e-04
Epoch 4/50
45/45 [==============================] - 5s 101ms/step - loss: 0.0950 - ac
c: 0.9697 - val_loss: 0.2239 - val_acc: 0.9253 - lr: 1.0000e-04
Epoch 5/50
45/45 [==============================] - 5s 91ms/step - loss: 0.0702 - acc
: 0.9807 - val_loss: 0.2247 - val_acc: 0.9218 - lr: 1.0000e-04
Epoch 6/50
45/45 [==============================] - 5s 88ms/step - loss: 0.0554 - acc
: 0.9866 - val_loss: 0.2258 - val_acc: 0.9263 - lr: 1.0000e-04
Epoch 7/50
45/45 [==============================] - 5s 89ms/step - loss: 0.0414 - acc
: 0.9914 - val_loss: 0.2431 - val_acc: 0.9243 - lr: 1.0000e-04
Epoch 8/50
45/45 [==============================] - 5s 89ms/step - loss: 0.0387 - acc
: 0.9921 - val_loss: 0.2370 - val_acc: 0.9273 - lr: 1.0000e-04
Epoch 9/50
45/45 [==============================] - 5s 90ms/step - loss: 0.0278 - acc
: 0.9962 - val_loss: 0.2512 - val_acc: 0.9248 - lr: 1.0000e-04
Epoch 10/50
45/45 [==============================] - 5s 93ms/step - loss: 0.0210 - acc
: 0.9982 - val_loss: 0.2449 - val_acc: 0.9258 - lr: 2.0000e-05
Epoch 11/50
45/45 [==============================] - 5s 91ms/step - loss: 0.0192 - acc
: 0.9983 - val_loss: 0.2463 - val_acc: 0.9268 - lr: 2.0000e-05
Epoch 12/50
45/45 [==============================] - 5s 90ms/step - loss: 0.0184 - acc
: 0.9988 - val_loss: 0.2480 - val_acc: 0.9273 - lr: 2.0000e-05
Epoch 13/50
45/45 [==============================] - 5s 92ms/step - loss: 0.0176 - acc
: 0.9988 - val_loss: 0.2489 - val_acc: 0.9273 - lr: 2.0000e-05
Epoch 14/50
45/45 [==============================] - 5s 91ms/step - loss: 0.0169 - acc
: 0.9987 - val_loss: 0.2500 - val_acc: 0.9278 - lr: 2.0000e-05
Final loss: training -> 0.02, validation -> 0.25
Final accuracy: training -> 1.00, validation -> 0.93
6/6 [==============================] - 1s 99ms/step - loss: 0.1698 - acc:
0.9432
Model evaluated: Test Loss-> 0.16977046430110931, Test Accuracy -> 94.32%
Model: "Xception-t"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_7 (Rescaling)     multiple                  0

 xception (Functional)       (None, 3, 3, 2048)        20861480

 flatten_4 (Flatten)         multiple                  0

 dense_13 (Dense)            multiple                  18433000

 dense_14 (Dense)            multiple                  5005
```

```
================================================================
Total params: 39299485 (149.92 MB)
Trainable params: 18438005 (70.34 MB)
Non-trainable params: 20861480 (79.58 MB)
________________________________________________________________
```

In [ ]:
```
plot_history(xception_t_history, ['loss', 'acc'], name=f'{xception_t.name
```



3.2. Fine-tuning

To finetune the pre-trained models we will first freeze the model and train only the "top", i.e. the classifier. After this, the whole model is unfrozen and trained for a few epochs with a very low learning rate in order to slightly modify the weights of our model and adapt it better to the dataset.

ResNet-50 V2

In [ ]:
```
resnet_f = PretrainedModel(num_classes=5, img_size=IMG_SIZE, pretrained='
optimizer = Adam(learning_rate=1e-3)
resnet_f.compile(optimizer=optimizer, loss="sparse_categorical_crossentro

print("Training only the classifier")
train_model(
    resnet_f,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
    lr_patience=3,
    val_patience=5,
)
```

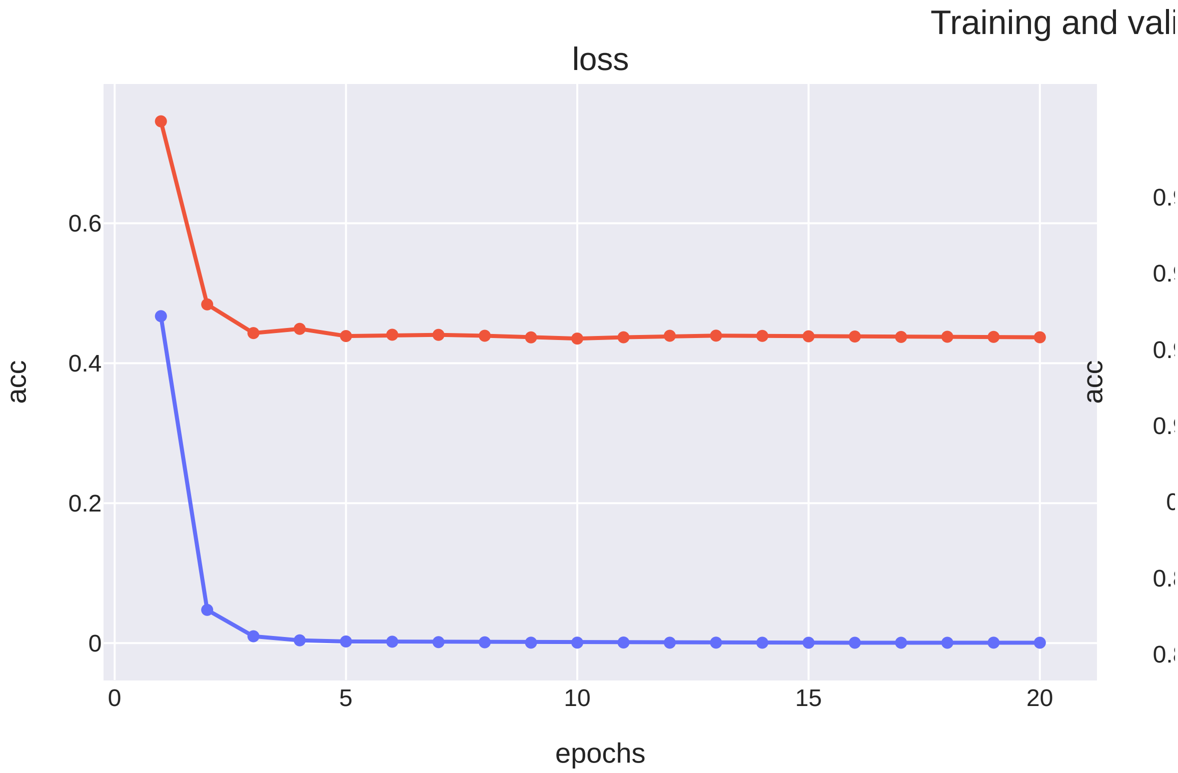
```
print("Finetuning the model with a learning rate of 1e-5")
resnet_f.trainable = True
optimizer = Adam(learning_rate=5e-5)
resnet_f.compile(optimizer=optimizer, loss="sparse_categorical_crossentro

resnet_f_history = train_model(
    resnet_f,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
    lr_patience=3,
    val_patience=10,
).history

print(
    f"Final loss: training -> {resnet_f_history['loss'][-1]:.2f}, validat
)
print(
    f"Final accuracy: training -> {resnet_f_history['acc'][-1]:.2f}, vali
)
test_loss, test_accuracy = resnet_f.evaluate(test_dataset, batch_size=BAT
print(
    f"Model evaluated: Test Loss-> {test_loss}, Test Accuracy -> {test_ac
)
resnet_f.summary()
```

```
Training only the classifier
Epoch 1/50
45/45 [==============================] - 7s 115ms/step - loss: 4.8100 - ac
c: 0.8467 - val_loss: 0.6332 - val_acc: 0.8936 - lr: 0.0010
Epoch 2/50
45/45 [==============================] - 6s 111ms/step - loss: 0.2195 - ac
c: 0.9454 - val_loss: 0.3765 - val_acc: 0.9124 - lr: 0.0010
Epoch 3/50
45/45 [==============================] - 6s 109ms/step - loss: 0.0665 - ac
c: 0.9804 - val_loss: 0.3454 - val_acc: 0.9124 - lr: 0.0010
Epoch 4/50
45/45 [==============================] - 5s 90ms/step - loss: 0.0256 - acc
: 0.9944 - val_loss: 0.3512 - val_acc: 0.9203 - lr: 0.0010
Epoch 5/50
45/45 [==============================] - 6s 108ms/step - loss: 0.0116 - ac
c: 0.9986 - val_loss: 0.3418 - val_acc: 0.9208 - lr: 0.0010
Epoch 6/50
45/45 [==============================] - 5s 91ms/step - loss: 0.0062 - acc
: 0.9995 - val_loss: 0.3484 - val_acc: 0.9238 - lr: 0.0010
Epoch 7/50
45/45 [==============================] - 5s 91ms/step - loss: 0.0044 - acc
: 0.9997 - val_loss: 0.3576 - val_acc: 0.9253 - lr: 0.0010
Epoch 8/50
45/45 [==============================] - 5s 91ms/step - loss: 0.0033 - acc
: 0.9997 - val_loss: 0.3649 - val_acc: 0.9258 - lr: 0.0010
Epoch 9/50
45/45 [==============================] - 5s 93ms/step - loss: 0.0021 - acc
: 0.9998 - val_loss: 0.3686 - val_acc: 0.9228 - lr: 2.0000e-04
Epoch 10/50
45/45 [==============================] - 5s 90ms/step - loss: 0.0020 - acc
: 0.9997 - val_loss: 0.3710 - val_acc: 0.9228 - lr: 2.0000e-04
Finetuning the model with a learning rate of 1e-5
Epoch 1/50
45/45 [==============================] - 27s 247ms/step - loss: 0.4672 - a
cc: 0.8630 - val_loss: 0.7456 - val_acc: 0.8906 - lr: 5.0000e-05
Epoch 2/50
45/45 [==============================] - 8s 160ms/step - loss: 0.0475 - ac
c: 0.9885 - val_loss: 0.4840 - val_acc: 0.9075 - lr: 5.0000e-05
Epoch 3/50
45/45 [==============================] - 8s 161ms/step - loss: 0.0098 - ac
c: 0.9990 - val_loss: 0.4431 - val_acc: 0.9060 - lr: 5.0000e-05
Epoch 4/50
45/45 [==============================] - 7s 137ms/step - loss: 0.0040 - ac
c: 0.9997 - val_loss: 0.4491 - val_acc: 0.9060 - lr: 5.0000e-05
Epoch 5/50
45/45 [==============================] - 8s 161ms/step - loss: 0.0023 - ac
c: 0.9997 - val_loss: 0.4388 - val_acc: 0.9050 - lr: 5.0000e-05
Epoch 6/50
45/45 [==============================] - 7s 137ms/step - loss: 0.0020 - ac
c: 0.9997 - val_loss: 0.4407 - val_acc: 0.9025 - lr: 5.0000e-05
Epoch 7/50
45/45 [==============================] - 7s 136ms/step - loss: 0.0014 - ac
c: 0.9998 - val_loss: 0.4405 - val_acc: 0.9035 - lr: 5.0000e-05
Epoch 8/50
45/45 [==============================] - 7s 137ms/step - loss: 0.0011 - ac
c: 0.9997 - val_loss: 0.4392 - val_acc: 0.9035 - lr: 5.0000e-05
Epoch 9/50
45/45 [==============================] - 8s 161ms/step - loss: 6.4849e-04
- acc: 0.9998 - val_loss: 0.4369 - val_acc: 0.9030 - lr: 1.0000e-05
Epoch 10/50
```

```
45/45 [==============================] - 8s 161ms/step - loss: 7.2401e-04
- acc: 0.9997 - val_loss: 0.4352 - val_acc: 0.8996 - lr: 1.0000e-05
Epoch 11/50
45/45 [==============================] - 7s 136ms/step - loss: 8.7058e-04
- acc: 0.9996 - val_loss: 0.4370 - val_acc: 0.8981 - lr: 1.0000e-05
Epoch 12/50
45/45 [==============================] - 7s 136ms/step - loss: 6.1167e-04
- acc: 0.9997 - val_loss: 0.4392 - val_acc: 0.8991 - lr: 1.0000e-05
Epoch 13/50
45/45 [==============================] - 7s 137ms/step - loss: 6.9154e-04
- acc: 0.9997 - val_loss: 0.4394 - val_acc: 0.8986 - lr: 1.0000e-05
Epoch 14/50
45/45 [==============================] - 7s 137ms/step - loss: 5.6919e-04
- acc: 0.9997 - val_loss: 0.4390 - val_acc: 0.8991 - lr: 2.0000e-06
Epoch 15/50
45/45 [==============================] - 7s 137ms/step - loss: 5.5034e-04
- acc: 0.9998 - val_loss: 0.4384 - val_acc: 0.8991 - lr: 2.0000e-06
Epoch 16/50
45/45 [==============================] - 7s 136ms/step - loss: 5.3281e-04
- acc: 0.9999 - val_loss: 0.4381 - val_acc: 0.8991 - lr: 2.0000e-06
Epoch 17/50
45/45 [==============================] - 7s 136ms/step - loss: 5.3672e-04
- acc: 0.9997 - val_loss: 0.4375 - val_acc: 0.8991 - lr: 4.0000e-07
Epoch 18/50
45/45 [==============================] - 7s 137ms/step - loss: 5.1441e-04
- acc: 0.9998 - val_loss: 0.4378 - val_acc: 0.8991 - lr: 4.0000e-07
Epoch 19/50
45/45 [==============================] - 7s 138ms/step - loss: 6.6209e-04
- acc: 0.9998 - val_loss: 0.4376 - val_acc: 0.8991 - lr: 4.0000e-07
Epoch 20/50
45/45 [==============================] - 7s 137ms/step - loss: 5.5975e-04
- acc: 0.9997 - val_loss: 0.4370 - val_acc: 0.8991 - lr: 8.0000e-08
Final loss: training -> 0.00, validation -> 0.44
Final accuracy: training -> 1.00, validation -> 0.90
6/6 [==============================] - 1s 49ms/step - loss: 0.3587 - acc:
0.9212
Model evaluated: Test Loss-> 0.3587038516998291, Test Accuracy -> 92.12%
Model: "ResNet-f"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_8 (Rescaling)     multiple                  0

 resnet50v2 (Functional)     (None, 4, 4, 2048)        23564800

 flatten_5 (Flatten)         multiple                  0

 dense_15 (Dense)            multiple                  32769000

 dense_16 (Dense)            multiple                  5005

=================================================================
Total params: 56338805 (214.92 MB)
Trainable params: 56293365 (214.74 MB)
Non-trainable params: 45440 (177.50 KB)
_________________________________________________________________
```

In [ ]:

```
plot_history(resnet_f_history, ['loss', 'acc'], name=f'{resnet_f.name} Mo
```



MobileNet V2

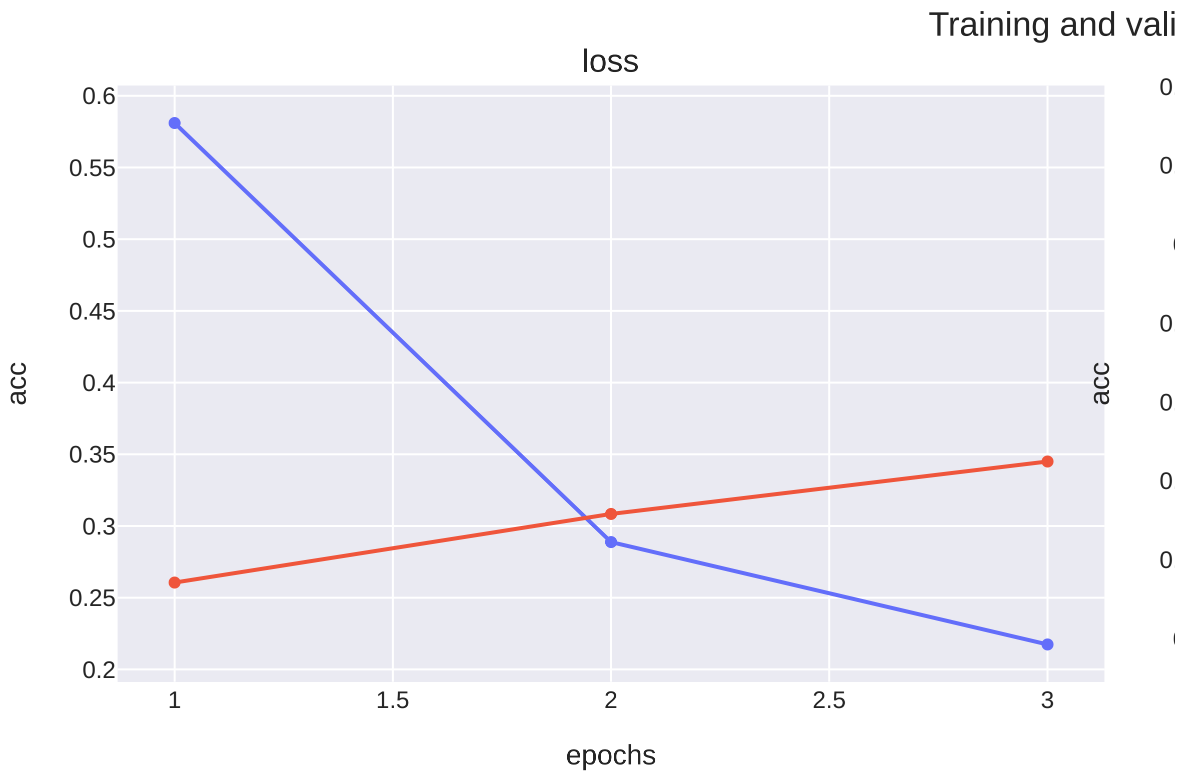
In [ ]:
```
mobile_f = PretrainedModel(num_classes=5, img_size=IMG_SIZE, pretrained='
optimizer = Adam(learning_rate=1e-3)
mobile_f.compile(optimizer=optimizer, loss="sparse_categorical_crossentro

print("Training only the classifier")
train_model(
    mobile_f,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
    lr_patience=3,
    val_patience=5,
)

print("Finetuning the model with a learning rate of 1e-5")
mobile_f.trainable = True
optimizer = Adam(learning_rate=1e-5)
mobile_f.compile(optimizer=optimizer, loss="sparse_categorical_crossentro

mobile_f_history = train_model(
    mobile_f,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
    lr_patience=2,
    val_patience=2,
).history
```

```
print(
    f"Final loss: training -> {mobile_f_history['loss'][-1]:.2f}, validat
)
print(
    f"Final accuracy: training -> {mobile_f_history['acc'][-1]:.2f}, vali
)
test_loss, test_accuracy = mobile_f.evaluate(test_dataset, batch_size=BAT
print(
    f"Model evaluated: Test Loss-> {test_loss}, Test Accuracy -> {test_ac
)
mobile_f.summary()
```

```
Training only the classifier
Epoch 1/50
45/45 [==============================] - 7s 102ms/step - loss: 8.0062 - ac
c: 0.8124 - val_loss: 0.2656 - val_acc: 0.9203 - lr: 0.0010
Epoch 2/50
45/45 [==============================] - 5s 98ms/step - loss: 0.1860 - acc
: 0.9368 - val_loss: 0.2131 - val_acc: 0.9312 - lr: 0.0010
Epoch 3/50
45/45 [==============================] - 5s 98ms/step - loss: 0.0900 - acc
: 0.9699 - val_loss: 0.1972 - val_acc: 0.9337 - lr: 0.0010
Epoch 4/50
45/45 [==============================] - 5s 87ms/step - loss: 0.0683 - acc
: 0.9766 - val_loss: 0.2179 - val_acc: 0.9381 - lr: 0.0010
Epoch 5/50
45/45 [==============================] - 5s 88ms/step - loss: 0.0386 - acc
: 0.9900 - val_loss: 0.1982 - val_acc: 0.9406 - lr: 0.0010
Epoch 6/50
45/45 [==============================] - 5s 87ms/step - loss: 0.0237 - acc
: 0.9966 - val_loss: 0.1981 - val_acc: 0.9401 - lr: 0.0010
Epoch 7/50
45/45 [==============================] - 5s 98ms/step - loss: 0.0149 - acc
: 0.9990 - val_loss: 0.1943 - val_acc: 0.9416 - lr: 2.0000e-04
Epoch 8/50
45/45 [==============================] - 5s 87ms/step - loss: 0.0133 - acc
: 0.9990 - val_loss: 0.1955 - val_acc: 0.9421 - lr: 2.0000e-04
Epoch 9/50
45/45 [==============================] - 5s 86ms/step - loss: 0.0122 - acc
: 0.9991 - val_loss: 0.1971 - val_acc: 0.9416 - lr: 2.0000e-04
Epoch 10/50
45/45 [==============================] - 5s 87ms/step - loss: 0.0112 - acc
: 0.9993 - val_loss: 0.1990 - val_acc: 0.9421 - lr: 2.0000e-04
Epoch 11/50
45/45 [==============================] - 5s 89ms/step - loss: 0.0102 - acc
: 0.9995 - val_loss: 0.1984 - val_acc: 0.9416 - lr: 4.0000e-05
Epoch 12/50
45/45 [==============================] - 5s 89ms/step - loss: 0.0100 - acc
: 0.9995 - val_loss: 0.1988 - val_acc: 0.9411 - lr: 4.0000e-05
Finetuning the model with a learning rate of 1e-5
Epoch 1/50
45/45 [==============================] - 20s 157ms/step - loss: 0.5810 - a
cc: 0.7985 - val_loss: 0.2606 - val_acc: 0.9307 - lr: 1.0000e-05
Epoch 2/50
45/45 [==============================] - 5s 93ms/step - loss: 0.2888 - acc
: 0.8928 - val_loss: 0.3084 - val_acc: 0.9253 - lr: 1.0000e-05
Epoch 3/50
45/45 [==============================] - 5s 92ms/step - loss: 0.2174 - acc
: 0.9205 - val_loss: 0.3450 - val_acc: 0.9208 - lr: 1.0000e-05
Final loss: training -> 0.22, validation -> 0.34
Final accuracy: training -> 0.92, validation -> 0.92
6/6 [==============================] - 0s 30ms/step - loss: 0.2889 - acc:
0.9352
Model evaluated: Test Loss-> 0.28893202543258667, Test Accuracy -> 93.52%
Model: "Mobile-f"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_9 (Rescaling)     multiple                  0

 mobilenetv2_1.00_224 (Func  (None, 4, 4, 1280)        2257984
 tional)
```

```
 flatten_6 (Flatten)          multiple                  0

 dense_17 (Dense)             multiple                  20481000

 dense_18 (Dense)             multiple                  5005

=================================================================
Total params: 22743989 (86.76 MB)
Trainable params: 22709877 (86.63 MB)
Non-trainable params: 34112 (133.25 KB)
_________________________________________________________________
```

In [ ]:
```
plot_history(mobile_f_history, ['loss', 'acc'], name=f'{mobile_f.name} Mo
```



Xception

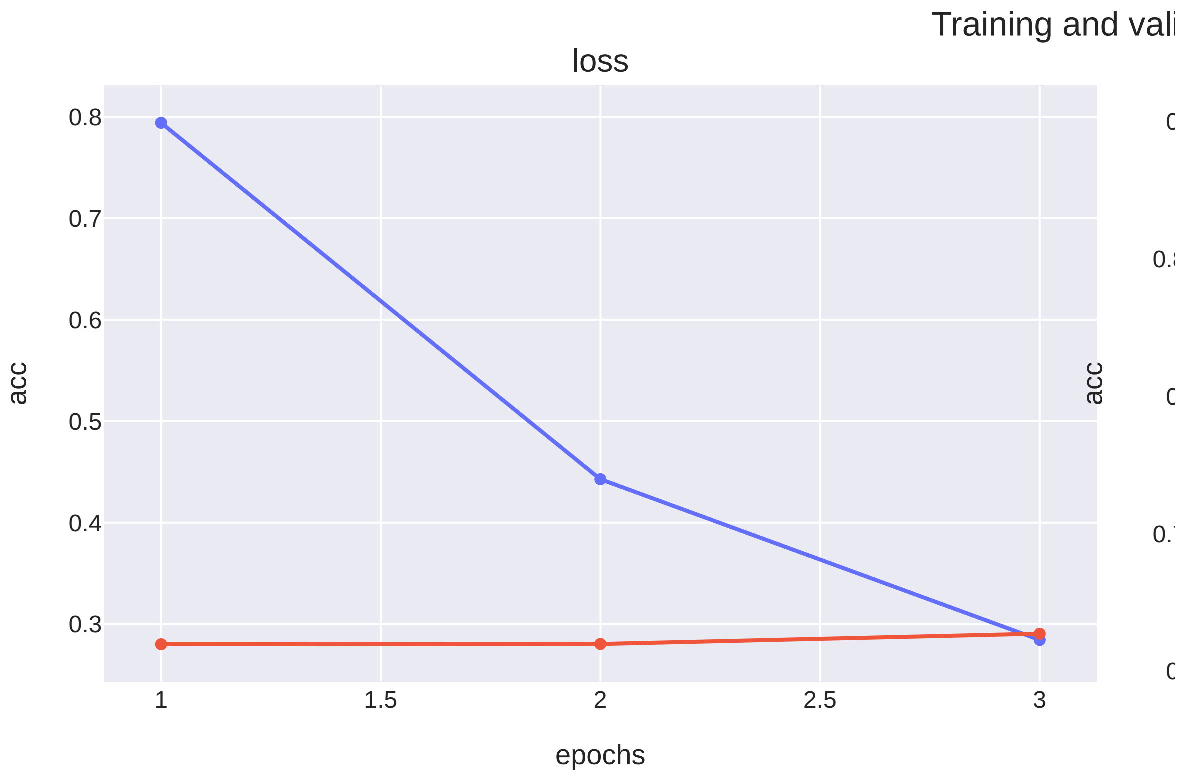
In [ ]:
```
xception_f = PretrainedModel(num_classes=5, img_size=IMG_SIZE, pretrained
optimizer = Adam(learning_rate=1e-3)
xception_f.compile(optimizer=optimizer, loss="sparse_categorical_crossent

print("Training only the classifier")
xception_f_history = train_model(
    xception_f,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
    lr_patience=3,
    val_patience=5,
).history

print("Finetuning the model with a learning rate of 1e-5")
xception_f.trainable = True
```

```
optimizer = Adam(learning_rate=1e-5)
xception_f.compile(optimizer=optimizer, loss="sparse_categorical_crossent

xception_f_history = train_model(
    xception_f,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
    lr_patience=2,
    val_patience=2,
).history

print(
    f"Final loss: training -> {xception_f_history['loss'][-1]:.2f}, valid
)
print(
    f"Final accuracy: training -> {xception_f_history['acc'][-1]:.2f}, va
)
test_loss, test_accuracy = xception_f.evaluate(test_dataset, batch_size=B
print(
    f"Model evaluated: Test Loss-> {test_loss}, Test Accuracy -> {test_ac
)
xception_f.summary()
```

```
Training only the classifier
Epoch 1/50
45/45 [==============================] - 7s 109ms/step - loss: 3.1739 - ac
c: 0.8092 - val_loss: 0.4876 - val_acc: 0.8936 - lr: 0.0010
Epoch 2/50
45/45 [==============================] - 6s 104ms/step - loss: 0.2881 - ac
c: 0.9221 - val_loss: 0.3135 - val_acc: 0.9065 - lr: 0.0010
Epoch 3/50
45/45 [==============================] - 6s 104ms/step - loss: 0.1798 - ac
c: 0.9403 - val_loss: 0.2606 - val_acc: 0.9184 - lr: 0.0010
Epoch 4/50
45/45 [==============================] - 5s 92ms/step - loss: 0.1568 - acc
: 0.9437 - val_loss: 0.2925 - val_acc: 0.9075 - lr: 0.0010
Epoch 5/50
45/45 [==============================] - 5s 92ms/step - loss: 0.1160 - acc
: 0.9564 - val_loss: 0.2681 - val_acc: 0.9134 - lr: 0.0010
Epoch 6/50
45/45 [==============================] - 5s 92ms/step - loss: 0.1087 - acc
: 0.9601 - val_loss: 0.2931 - val_acc: 0.9174 - lr: 0.0010
Epoch 7/50
45/45 [==============================] - 5s 90ms/step - loss: 0.0717 - acc
: 0.9761 - val_loss: 0.2623 - val_acc: 0.9144 - lr: 2.0000e-04
Epoch 8/50
45/45 [==============================] - 6s 103ms/step - loss: 0.0602 - ac
c: 0.9828 - val_loss: 0.2582 - val_acc: 0.9149 - lr: 2.0000e-04
Epoch 9/50
45/45 [==============================] - 5s 93ms/step - loss: 0.0555 - acc
: 0.9854 - val_loss: 0.2591 - val_acc: 0.9154 - lr: 2.0000e-04
Epoch 10/50
45/45 [==============================] - 5s 91ms/step - loss: 0.0529 - acc
: 0.9866 - val_loss: 0.2611 - val_acc: 0.9179 - lr: 2.0000e-04
Epoch 11/50
45/45 [==============================] - 5s 94ms/step - loss: 0.0475 - acc
: 0.9896 - val_loss: 0.2585 - val_acc: 0.9189 - lr: 2.0000e-04
Epoch 12/50
45/45 [==============================] - 6s 102ms/step - loss: 0.0426 - ac
c: 0.9909 - val_loss: 0.2576 - val_acc: 0.9164 - lr: 4.0000e-05
Epoch 13/50
45/45 [==============================] - 5s 90ms/step - loss: 0.0417 - acc
: 0.9917 - val_loss: 0.2580 - val_acc: 0.9174 - lr: 4.0000e-05
Epoch 14/50
45/45 [==============================] - 5s 92ms/step - loss: 0.0411 - acc
: 0.9920 - val_loss: 0.2582 - val_acc: 0.9169 - lr: 4.0000e-05
Epoch 15/50
45/45 [==============================] - 5s 90ms/step - loss: 0.0401 - acc
: 0.9921 - val_loss: 0.2586 - val_acc: 0.9164 - lr: 4.0000e-05
Epoch 16/50
45/45 [==============================] - 5s 91ms/step - loss: 0.0393 - acc
: 0.9924 - val_loss: 0.2584 - val_acc: 0.9203 - lr: 8.0000e-06
Epoch 17/50
45/45 [==============================] - 5s 91ms/step - loss: 0.0388 - acc
: 0.9926 - val_loss: 0.2585 - val_acc: 0.9198 - lr: 8.0000e-06
Finetuning the model with a learning rate of 1e-5
Epoch 1/50
45/45 [==============================] - 25s 237ms/step - loss: 0.7941 - a
cc: 0.7098 - val_loss: 0.2801 - val_acc: 0.8996 - lr: 1.0000e-05
Epoch 2/50
45/45 [==============================] - 8s 153ms/step - loss: 0.4428 - ac
c: 0.8434 - val_loss: 0.2804 - val_acc: 0.8971 - lr: 1.0000e-05
Epoch 3/50
```

```
45/45 [==============================] - 8s 154ms/step - loss: 0.2844 - ac
c: 0.8967 - val_loss: 0.2906 - val_acc: 0.8926 - lr: 1.0000e-05
Final loss: training -> 0.28, validation -> 0.29
Final accuracy: training -> 0.90, validation -> 0.89
6/6 [==============================] - 1s 51ms/step - loss: 0.2098 - acc:
0.9185
Model evaluated: Test Loss-> 0.20984682440757751, Test Accuracy -> 91.85%
Model: "Xception-f"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_10 (Rescaling)    multiple                  0

 xception (Functional)       (None, 3, 3, 2048)        20861480

 flatten_7 (Flatten)         multiple                  0

 dense_19 (Dense)            multiple                  18433000

 dense_20 (Dense)            multiple                  5005

=================================================================
Total params: 39299485 (149.92 MB)
Trainable params: 39244957 (149.71 MB)
Non-trainable params: 54528 (213.00 KB)
_________________________________________________________________
```

In [ ]:
```
plot_history(xception_f_history, ['loss', 'acc'], name=f'{xception_f.name
```



3.3. Partial fine-tuning

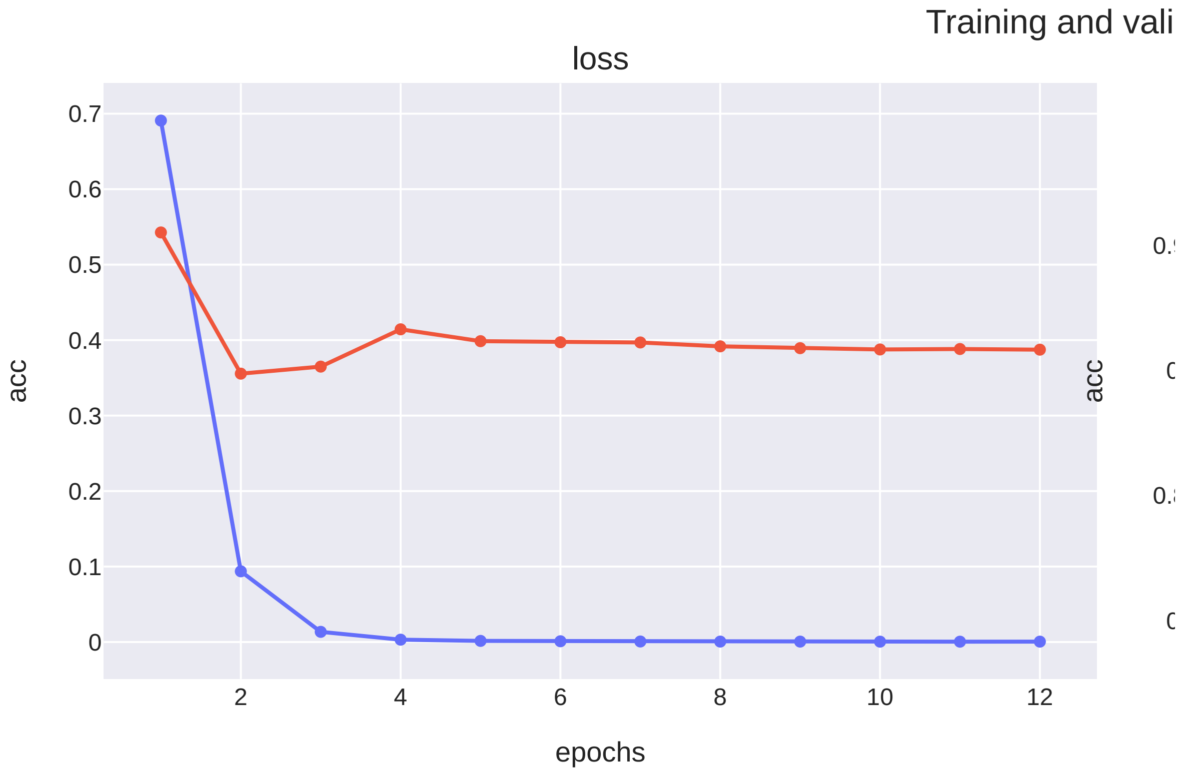
ResNet-50 V2

In [ ]:
```
resnet_d = PretrainedModel(num_classes=5, img_size=IMG_SIZE, pretrained='
```

```
optimizer = Adam(learning_rate=1e-4)
resnet_d.compile(optimizer=optimizer, loss="sparse_categorical_crossentro
resnet_d_history = train_model(
    resnet_d,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
    lr_patience=5,
    val_patience=10
).history
print(
    f"Final loss: training -> {resnet_d_history['loss'][-1]:.2f}, validat
)
print(
    f"Final accuracy: training -> {resnet_d_history['acc'][-1]:.2f}, vali
)
test_loss, test_accuracy = resnet_d.evaluate(test_dataset, batch_size=BAT
print(
    f"Model evaluated: Test Loss-> {test_loss}, Test Accuracy -> {test_ac
)
resnet_d.summary()
```

```
Epoch 1/50
45/45 [==============================] - 18s 168ms/step - loss: 0.6909 - a
cc: 0.7916 - val_loss: 0.5427 - val_acc: 0.8946 - lr: 1.0000e-04
Epoch 2/50
45/45 [==============================] - 8s 159ms/step - loss: 0.0937 - ac
c: 0.9708 - val_loss: 0.3556 - val_acc: 0.9164 - lr: 1.0000e-04
Epoch 3/50
45/45 [==============================] - 7s 135ms/step - loss: 0.0135 - ac
c: 0.9967 - val_loss: 0.3649 - val_acc: 0.9218 - lr: 1.0000e-04
Epoch 4/50
45/45 [==============================] - 7s 136ms/step - loss: 0.0031 - ac
c: 0.9994 - val_loss: 0.4143 - val_acc: 0.9159 - lr: 1.0000e-04
Epoch 5/50
45/45 [==============================] - 7s 137ms/step - loss: 0.0015 - ac
c: 0.9997 - val_loss: 0.3986 - val_acc: 0.9208 - lr: 1.0000e-04
Epoch 6/50
45/45 [==============================] - 7s 135ms/step - loss: 0.0010 - ac
c: 0.9997 - val_loss: 0.3972 - val_acc: 0.9213 - lr: 1.0000e-04
Epoch 7/50
45/45 [==============================] - 7s 135ms/step - loss: 7.2656e-04
- acc: 0.9997 - val_loss: 0.3969 - val_acc: 0.9198 - lr: 1.0000e-04
Epoch 8/50
45/45 [==============================] - 7s 135ms/step - loss: 6.2142e-04
- acc: 0.9998 - val_loss: 0.3918 - val_acc: 0.9184 - lr: 2.0000e-05
Epoch 9/50
45/45 [==============================] - 7s 135ms/step - loss: 5.8788e-04
- acc: 0.9997 - val_loss: 0.3894 - val_acc: 0.9189 - lr: 2.0000e-05
Epoch 10/50
45/45 [==============================] - 7s 136ms/step - loss: 5.1186e-04
- acc: 0.9997 - val_loss: 0.3876 - val_acc: 0.9184 - lr: 2.0000e-05
Epoch 11/50
45/45 [==============================] - 7s 136ms/step - loss: 4.7585e-04
- acc: 0.9997 - val_loss: 0.3882 - val_acc: 0.9169 - lr: 2.0000e-05
Epoch 12/50
45/45 [==============================] - 7s 132ms/step - loss: 5.2060e-04
- acc: 0.9997 - val_loss: 0.3874 - val_acc: 0.9169 - lr: 2.0000e-05
Final loss: training -> 0.00, validation -> 0.39
Final accuracy: training -> 1.00, validation -> 0.92
6/6 [==============================] - 1s 52ms/step - loss: 0.3425 - acc:
0.9279
Model evaluated: Test Loss-> 0.3424745500087738, Test Accuracy -> 92.79%
Model: "ResNet-d"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_11 (Rescaling)    multiple                  0

 resnet50v2 (Functional)     (None, 4, 4, 2048)        23564800

 flatten_8 (Flatten)         multiple                  0

 dense_21 (Dense)            multiple                  32769000

 dense_22 (Dense)            multiple                  5005

=================================================================
Total params: 56338805 (214.92 MB)
Trainable params: 52877301 (201.71 MB)
Non-trainable params: 3461504 (13.20 MB)
_________________________________________________________________
```

In [ ]:
```
plot_history(resnet_d_history, ['loss', 'acc'], name=f'{resnet_d.name} Mo
```

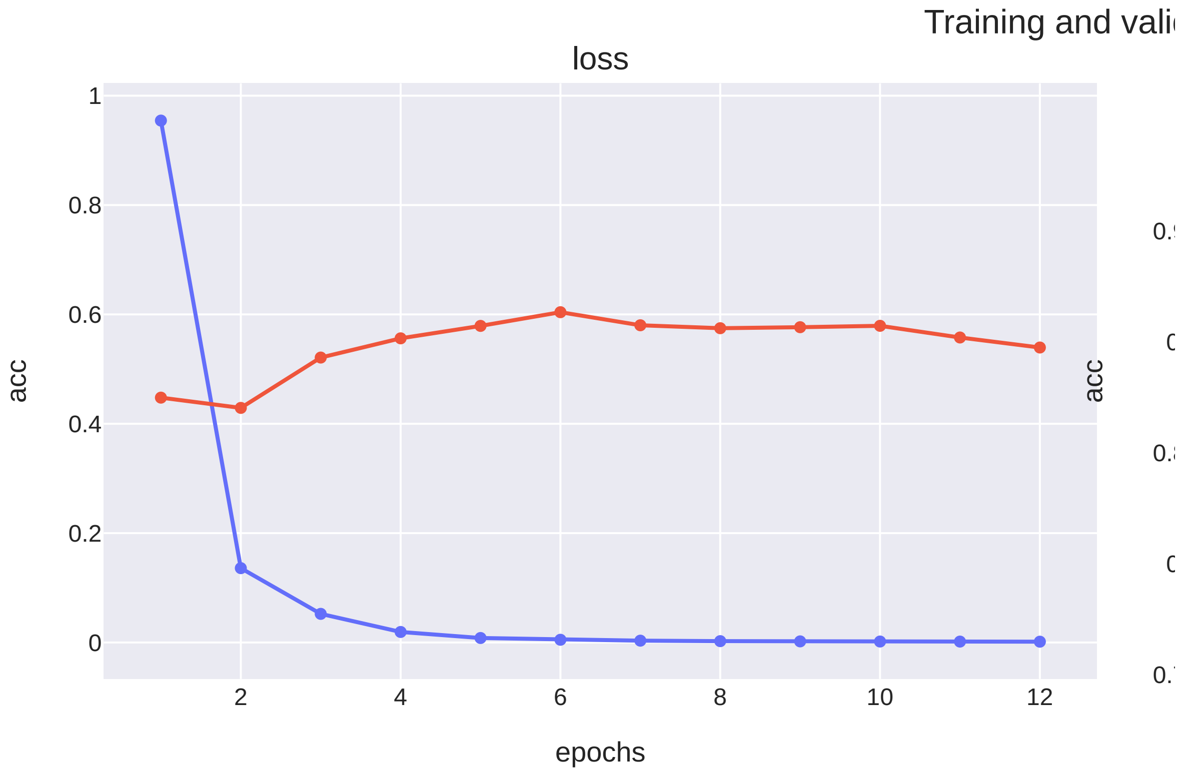


MobileNet V2

In [ ]:
```
mobile_d = PretrainedModel(num_classes=5, img_size=IMG_SIZE, pretrained='
optimizer = Adam(learning_rate=1e-4)
mobile_d.compile(optimizer=optimizer, loss="sparse_categorical_crossentro
mobile_d_history = train_model(
    mobile_d,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
    lr_patience=5,
    val_patience=10
).history
print(
    f"Final loss: training -> {mobile_d_history['loss'][-1]:.2f}, validat
)
print(
    f"Final accuracy: training -> {mobile_d_history['acc'][-1]:.2f}, vali
)
test_loss, test_accuracy = mobile_d.evaluate(test_dataset, batch_size=BAT
print(
    f"Model evaluated: Test Loss-> {test_loss}, Test Accuracy -> {test_ac
)
mobile_d.summary()
```

```
Epoch 1/50
45/45 [==============================] - 14s 112ms/step - loss: 0.9545 - a
cc: 0.7648 - val_loss: 0.4478 - val_acc: 0.8822 - lr: 1.0000e-04
Epoch 2/50
45/45 [==============================] - 6s 113ms/step - loss: 0.1360 - ac
c: 0.9521 - val_loss: 0.4290 - val_acc: 0.8966 - lr: 1.0000e-04
Epoch 3/50
45/45 [==============================] - 5s 91ms/step - loss: 0.0524 - acc
: 0.9856 - val_loss: 0.5211 - val_acc: 0.8916 - lr: 1.0000e-04
Epoch 4/50
45/45 [==============================] - 5s 92ms/step - loss: 0.0193 - acc
: 0.9974 - val_loss: 0.5562 - val_acc: 0.8951 - lr: 1.0000e-04
Epoch 5/50
45/45 [==============================] - 5s 92ms/step - loss: 0.0082 - acc
: 0.9994 - val_loss: 0.5790 - val_acc: 0.8966 - lr: 1.0000e-04
Epoch 6/50
45/45 [==============================] - 5s 90ms/step - loss: 0.0050 - acc
: 0.9997 - val_loss: 0.6040 - val_acc: 0.8976 - lr: 1.0000e-04
Epoch 7/50
45/45 [==============================] - 5s 92ms/step - loss: 0.0033 - acc
: 0.9997 - val_loss: 0.5802 - val_acc: 0.9015 - lr: 1.0000e-04
Epoch 8/50
45/45 [==============================] - 5s 90ms/step - loss: 0.0025 - acc
: 0.9998 - val_loss: 0.5748 - val_acc: 0.9010 - lr: 2.0000e-05
Epoch 9/50
45/45 [==============================] - 5s 91ms/step - loss: 0.0020 - acc
: 0.9999 - val_loss: 0.5766 - val_acc: 0.9010 - lr: 2.0000e-05
Epoch 10/50
45/45 [==============================] - 5s 93ms/step - loss: 0.0018 - acc
: 0.9997 - val_loss: 0.5791 - val_acc: 0.9015 - lr: 2.0000e-05
Epoch 11/50
45/45 [==============================] - 5s 92ms/step - loss: 0.0017 - acc
: 0.9998 - val_loss: 0.5578 - val_acc: 0.9045 - lr: 2.0000e-05
Epoch 12/50
45/45 [==============================] - 5s 92ms/step - loss: 0.0015 - acc
: 0.9998 - val_loss: 0.5395 - val_acc: 0.9065 - lr: 2.0000e-05
Final loss: training -> 0.00, validation -> 0.54
Final accuracy: training -> 1.00, validation -> 0.91
6/6 [==============================] - 0s 29ms/step - loss: 0.3581 - acc:
0.9359
Model evaluated: Test Loss-> 0.3581494092941284, Test Accuracy -> 93.59%
Model: "Mobile-d"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_12 (Rescaling)    multiple                  0

 mobilenetv2_1.00_224 (Func  (None, 4, 4, 1280)        2257984
 tional)

 flatten_9 (Flatten)         multiple                  0

 dense_23 (Dense)            multiple                  20481000

 dense_24 (Dense)            multiple                  5005

=================================================================
Total params: 22743989 (86.76 MB)
Trainable params: 21977397 (83.84 MB)
Non-trainable params: 766592 (2.92 MB)
```

In [ ]:
```
plot_history(mobile_d_history, ['loss', 'acc'], name=f'{mobile_d.name} Mo
```



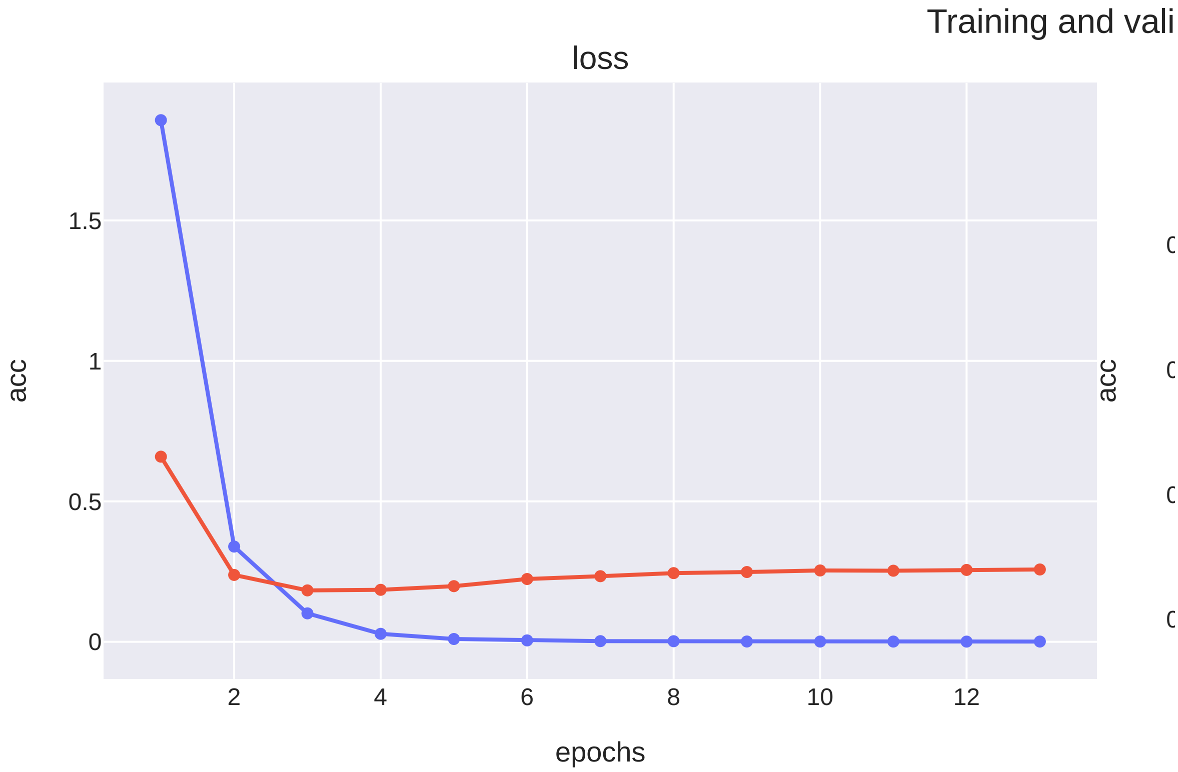
Xception

In [ ]:
```
xception_d = PretrainedModel(num_classes=5, img_size=IMG_SIZE, pretrained
optimizer = Adam(learning_rate=1e-4)
xception_d.compile(optimizer=optimizer, loss="sparse_categorical_crossent
xception_d_history = train_model(
    xception_d,
    train_dataset,
    val_dataset,
    path=model_dir,
    epochs=50,
    verbose=1,
    lr_patience=5,
    val_patience=10
).history
print(
    f"Final loss: training -> {xception_d_history['loss'][-1]:.2f}, valid
)
print(
    f"Final accuracy: training -> {xception_d_history['acc'][-1]:.2f}, va
)
test_loss, test_accuracy = xception_d.evaluate(test_dataset, batch_size=B
print(
    f"Model evaluated: Test Loss-> {test_loss}, Test Accuracy -> {test_ac
)
xception_d.summary()
```

```
Epoch 1/50
45/45 [==============================] - 16s 170ms/step - loss: 1.8568 - a
cc: 0.5821 - val_loss: 0.6590 - val_acc: 0.7848 - lr: 1.0000e-04
Epoch 2/50
45/45 [==============================] - 8s 163ms/step - loss: 0.3390 - ac
c: 0.8896 - val_loss: 0.2381 - val_acc: 0.9144 - lr: 1.0000e-04
Epoch 3/50
45/45 [==============================] - 8s 164ms/step - loss: 0.1013 - ac
c: 0.9652 - val_loss: 0.1830 - val_acc: 0.9391 - lr: 1.0000e-04
Epoch 4/50
45/45 [==============================] - 8s 148ms/step - loss: 0.0284 - ac
c: 0.9921 - val_loss: 0.1851 - val_acc: 0.9426 - lr: 1.0000e-04
Epoch 5/50
45/45 [==============================] - 8s 147ms/step - loss: 0.0101 - ac
c: 0.9975 - val_loss: 0.1981 - val_acc: 0.9436 - lr: 1.0000e-04
Epoch 6/50
45/45 [==============================] - 8s 146ms/step - loss: 0.0051 - ac
c: 0.9989 - val_loss: 0.2235 - val_acc: 0.9461 - lr: 1.0000e-04
Epoch 7/50
45/45 [==============================] - 8s 147ms/step - loss: 0.0024 - ac
c: 0.9996 - val_loss: 0.2336 - val_acc: 0.9461 - lr: 1.0000e-04
Epoch 8/50
45/45 [==============================] - 8s 146ms/step - loss: 0.0021 - ac
c: 0.9996 - val_loss: 0.2445 - val_acc: 0.9471 - lr: 1.0000e-04
Epoch 9/50
45/45 [==============================] - 8s 147ms/step - loss: 0.0011 - ac
c: 0.9998 - val_loss: 0.2484 - val_acc: 0.9476 - lr: 2.0000e-05
Epoch 10/50
45/45 [==============================] - 8s 147ms/step - loss: 9.8106e-04
- acc: 0.9997 - val_loss: 0.2540 - val_acc: 0.9461 - lr: 2.0000e-05
Epoch 11/50
45/45 [==============================] - 8s 147ms/step - loss: 8.8258e-04
- acc: 0.9997 - val_loss: 0.2531 - val_acc: 0.9461 - lr: 2.0000e-05
Epoch 12/50
45/45 [==============================] - 8s 147ms/step - loss: 7.6186e-04
- acc: 0.9997 - val_loss: 0.2559 - val_acc: 0.9461 - lr: 2.0000e-05
Epoch 13/50
45/45 [==============================] - 8s 146ms/step - loss: 8.8458e-04
- acc: 0.9997 - val_loss: 0.2575 - val_acc: 0.9456 - lr: 2.0000e-05
Final loss: training -> 0.00, validation -> 0.26
Final accuracy: training -> 1.00, validation -> 0.95
6/6 [==============================] - 1s 50ms/step - loss: 0.2511 - acc:
0.9486
Model evaluated: Test Loss-> 0.2511465847492218, Test Accuracy -> 94.86%
Model: "Xception-d"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_13 (Rescaling)    multiple                  0

 xception (Functional)       (None, 3, 3, 2048)        20861480

 flatten_10 (Flatten)        multiple                  0

 dense_25 (Dense)            multiple                  18433000

 dense_26 (Dense)            multiple                  5005

=================================================================
Total params: 39299485 (149.92 MB)
```

```
Trainable params: 33748637 (128.74 MB)
Non-trainable params: 5550848 (21.17 MB)
```

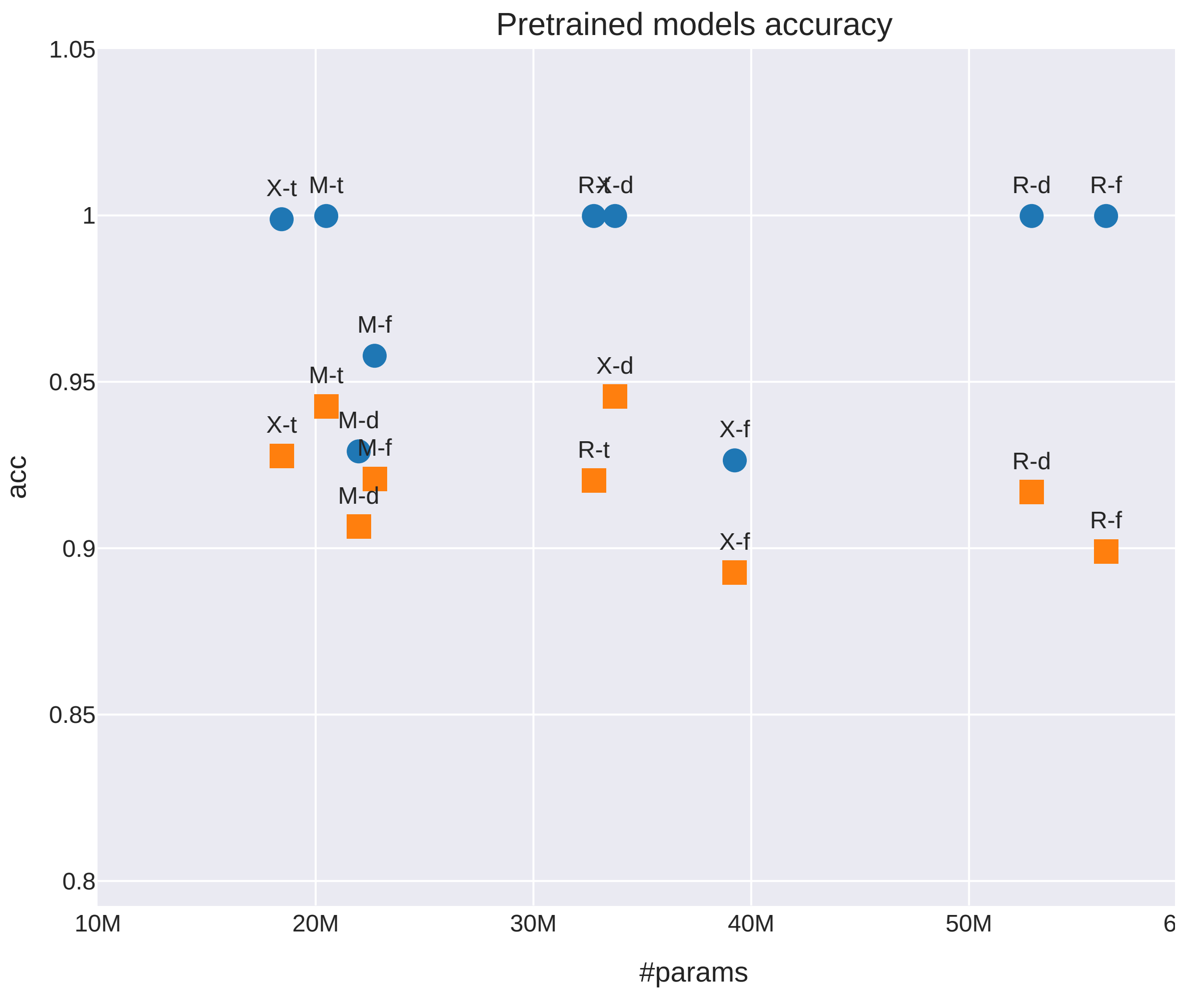
```
In [ ]: plot_history(xception_d_history, ['loss', 'acc'], name=f'{xception_d.name
```



3.4. Comparison between pretrained approaches

The next figure shows the train and test performance of each pretrained model with the number of trainable parameters. We have abbreviated the names with the first character and a suffix indicating the transfer-learning (t), finetuning (f) and partial-finetuning (d) approaches.

```
In [ ]: # abbreviate names
names = ['R-t', 'R-f', 'R-d', 'M-t', 'M-f', 'M-d', 'X-t', 'X-f', 'X-d']
models = [resnet_t, resnet_f, resnet_d, mobile_t, mobile_f, mobile_d, xce
for model, name in zip(models, names):
    model._name = name
comparison(models, (train_dataset, val_dataset), titles=['Pretrained mode
```

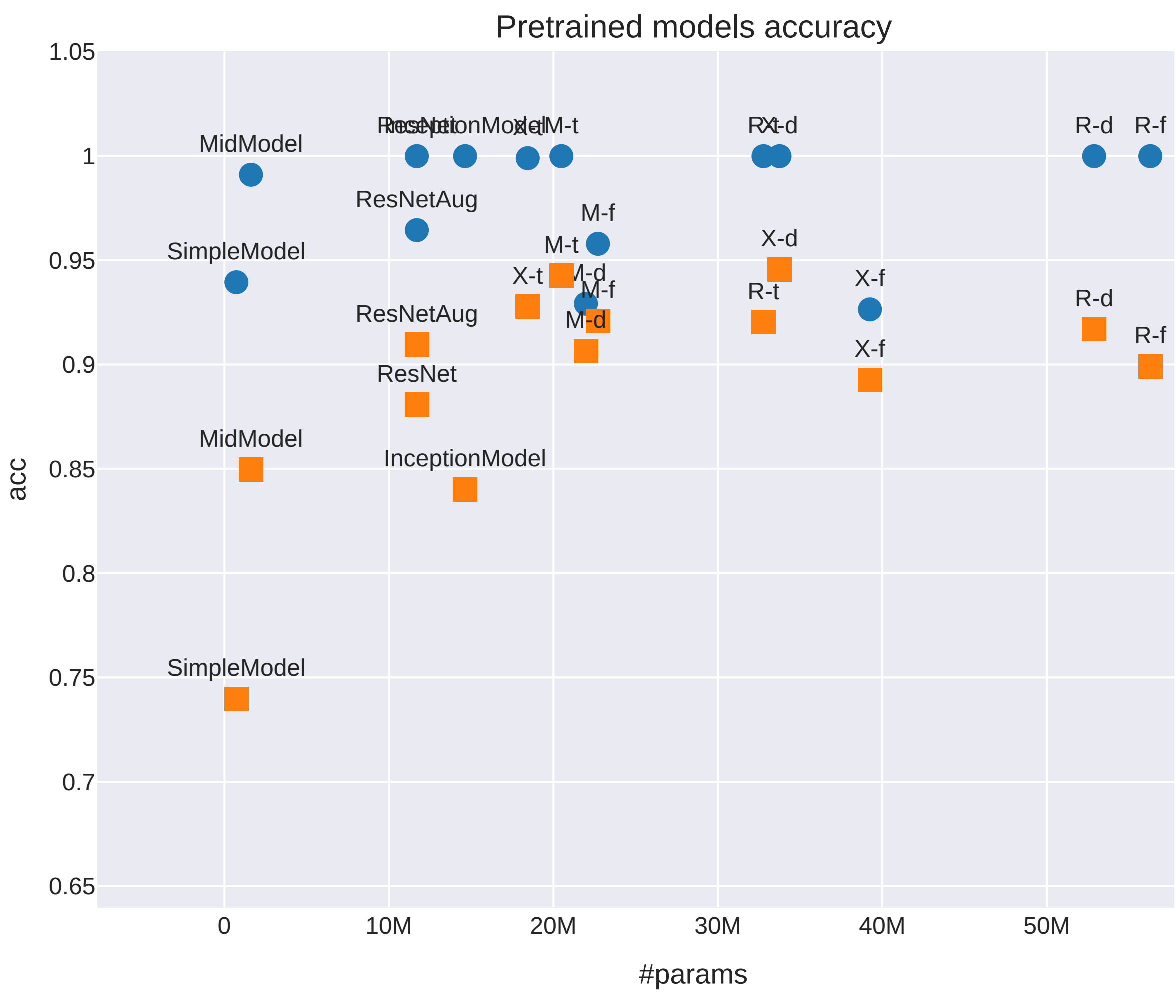


We can derive some conclusions about the above figure:

1. The best performance is achieved with the MobileNetV2 with transfer learning and Xception with partial fine-tuning.
2. In general terms, MobileNetV2 retrieves the better results, followed by Xception and finally ResNet50-V2.
3. Fine-tuning with Xception and ResNet50-V2 obtains the worst results (less than 90% of accuracy).
4. Although the number of trainable parameters is higher in the fine-tuning approaches, they yield the less overfitting results. This behavior might be due to the difference between the original pretraining image resolution (224×224) and the image resolution used for the animals dataset. Forcing the first layers to maintain the same weights might cause the models to skip some local features that are used genearlize the classification information.

4. Results and comparison

In [ ]:
```
models = [
    simple_model, mid_model, inception, resnet, resnet_data_aug,
    resnet_t, resnet_f, resnet_d, mobile_t, mobile_f, mobile_d, xception_
]


comparison(models, (train_dataset, val_dataset), titles=['Pretrained mode
```



The figure above shows a final comparison of all models trained in this notebook. In general terms, the pretrained architectures retrieve the best performance and overfitting level. Although the pretrained architectures are larger than the custom models, they can maintain the accuracy and generalization (avoiding overfitting) for encoding a lot of image information in their hidden layers and use a smaller learning rate to finetune them. Only the ResNet custom model is able to reach a lower overfitting level (similar to pretrained models): this success is obtained due to the data augmentation techinique, which simulates the pretraining process by allowing the model to see a large number of diverse images.