In [ ]:

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:

```
import tensorflow as tf
from tensorflow import keras
```

Loading Images

In [ ]:

```
from tensorflow.keras.preprocessing.image import load_img
```

In [ ]:

```
path = '../input/clothing-dataset-small/train/t-shirt'
name = '5f0a3fa0-6a3d-4b68-b213-72766a643de7.jpg'
fullname = path + '/' + name
load_img(fullname)
```

Out[ ]:

Usually we resize images. This is how a network will see these images:

In [ ]:

```
load_img(fullname, target_size=(299, 299))
```

Out[ ]:

In [ ]:

```
load_img(fullname, target_size=(150, 150))
```

Out[ ]:

Pre-Trained Neural Network

Let's apply a pre-trained neural network with imagenet classes.

We'll use Xception.

In [ ]:

```
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.applications.xception import preprocess_input
from tensorflow.keras.applications.xception import decode_predictions
```

Next,

- we load the image using the `load_img` function
- convert it to a numpy array
- make it a batch of one example

In [ ]:

```
img = load_img(fullname, target_size=(299, 299))
x = np.array(img)
x.shape
```

Out[ ]:

```
(299, 299, 3)
```

In [ ]:

```
X = np.array([x])
X.shape
```

```
(1, 299, 299, 3)
```

Out[ ]:

We're ready!

Next, we will:

- prepare the input
- do the predictions
- convert the predictions into a human-readable format

In [ ]:

```
X = preprocess_input(X)
```

Transfer learning

Instead of loading each image one-by-one, we can use a data generator. Keras will use it for loading the images and pre-processing them

In [ ]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

We'll use smaller images - it'll be faster

In [ ]:

```
image_size = (150, 150)
batch_size = 32
```

Let's get train data:

In [ ]:

```
train_gen = ImageDataGenerator(preprocessing_function=preprocess_input)

train_ds = train_gen.flow_from_directory(
    "../input/clothing-dataset-small/train",
    seed=1,
    target_size=image_size,
    batch_size=batch_size,
)
```

```
Found 3068 images belonging to 10 classes.
```

And validation:

In [ ]:

```
validation_gen = ImageDataGenerator(preprocessing_function=preprocess_input)

val_ds = validation_gen.flow_from_directory(
    "../input/clothing-dataset-small/validation",
    seed=1,
    target_size=image_size,
    batch_size=batch_size,
)
```

```
Found 341 images belonging to 10 classes.
```

For fine-tuning, we'll use `Xception` with small images (150x150)

In [ ]:

```
base_model = Xception(
    weights='imagenet',
    input_shape=(150, 150, 3),
    include_top=False
```

```
)

base_model.trainable = False
```

```
2021-12-12 06:28:50.987677: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:9
37] successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2021-12-12 06:28:51.084980: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:9
37] successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2021-12-12 06:28:51.086029: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:9
37] successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2021-12-12 06:28:51.088588: I tensorflow/core/platform/cpu_feature_guard.cc:142] Thi
s TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to
use the following CPU instructions in performance-critical operations:  AVX2 AVX512F
FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler
flags.
2021-12-12 06:28:51.089745: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:9
37] successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2021-12-12 06:28:51.090999: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:9
37] successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2021-12-12 06:28:51.092077: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:9
37] successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2021-12-12 06:28:52.888258: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:9
37] successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2021-12-12 06:28:52.889147: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:9
37] successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2021-12-12 06:28:52.890023: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:9
37] successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2021-12-12 06:28:52.890700: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510]
Created device /job:localhost/replica:0/task:0/device:GPU:0 with 15403 MB memory:  -
> device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capabilit
y: 6.0
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/x
ception/xception_weights_tf_dim_ordering_tf_kernels_notop.h5
83689472/83683744 [==============================] - 3s 0us/step
83697664/83683744 [==============================] - 3s 0us/step
```

Let's add a small neural net on top of that: just one layer with 10 neurons (there are 10 classes we want to predict)

In [ ]:

```
inputs = keras.Input(shape=(150, 150, 3))

base = base_model(inputs, training=False)
vector = keras.layers.GlobalAveragePooling2D()(base)
outputs = keras.layers.Dense(10)(vector)

model = keras.Model(inputs, outputs)
```

Now we specify the learning rate and compile the model. After that, it's ready for training

In [ ]:

```
learning_rate = 0.01

model.compile(
```

```
    optimizer=keras.optimizers.Adam(learning_rate),
    loss=keras.losses.CategoricalCrossentropy(from_logits=True),
    metrics=["accuracy"],
)
```

Let's train now for 10 epochs:

In [ ]:

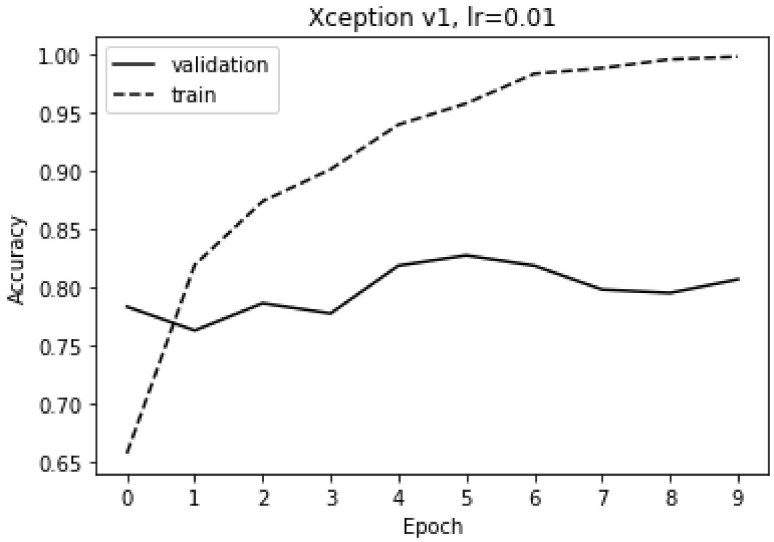
```
history = model.fit(train_ds, epochs=10, validation_data=val_ds)
```

```
2021-12-12 06:28:59.861560: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.
cc:185] None of the MLIR Optimization Passes are enabled (registered 2)
Epoch 1/10
2021-12-12 06:29:03.173851: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loade
d cuDNN version 8005
96/96 [==============================] - 33s 252ms/step - loss: 1.3167 - accuracy:
0.6568 - val_loss: 0.8293 - val_accuracy: 0.7830
Epoch 2/10
96/96 [==============================] - 14s 150ms/step - loss: 0.6025 - accuracy:
0.8188 - val_loss: 0.8861 - val_accuracy: 0.7625
Epoch 3/10
96/96 [==============================] - 15s 154ms/step - loss: 0.3709 - accuracy:
0.8735 - val_loss: 0.9536 - val_accuracy: 0.7859
Epoch 4/10
96/96 [==============================] - 14s 146ms/step - loss: 0.2840 - accuracy:
0.9009 - val_loss: 0.9727 - val_accuracy: 0.7771
Epoch 5/10
96/96 [==============================] - 14s 148ms/step - loss: 0.1643 - accuracy:
0.9394 - val_loss: 0.8617 - val_accuracy: 0.8182
Epoch 6/10
96/96 [==============================] - 14s 142ms/step - loss: 0.1263 - accuracy:
0.9576 - val_loss: 0.8038 - val_accuracy: 0.8270
Epoch 7/10
96/96 [==============================] - 14s 144ms/step - loss: 0.0578 - accuracy:
0.9831 - val_loss: 0.7822 - val_accuracy: 0.8182
Epoch 8/10
96/96 [==============================] - 15s 151ms/step - loss: 0.0419 - accuracy:
0.9879 - val_loss: 0.8748 - val_accuracy: 0.7977
Epoch 9/10
96/96 [==============================] - 14s 144ms/step - loss: 0.0264 - accuracy:
0.9954 - val_loss: 0.8370 - val_accuracy: 0.7947
Epoch 10/10
96/96 [==============================] - 15s 154ms/step - loss: 0.0235 - accuracy:
0.9977 - val_loss: 0.8383 - val_accuracy: 0.8065
```

In [ ]:

```
plt.figure(figsize=(6, 4))

epochs = history.epoch
val = history.history['val_accuracy']
train = history.history['accuracy']

plt.plot(epochs, val, color='black', linestyle='solid', label='validation')
plt.plot(epochs, train, color='black', linestyle='dashed', label='train')

plt.title('Xception v1, lr=0.01')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')

plt.xticks(np.arange(10))

plt.legend()
```

```
plt.savefig('xception_v1_0_01.svg')

plt.show()
```



0.01 is not necessarily the best learning rate, so we should experiment with 0.001.

To make it easier for us, let's make a function for defining our model:

In [ ]:

```
def make_model(learning_rate):
    base_model = Xception(
        weights='imagenet',
        input_shape=(150, 150, 3),
        include_top=False
    )

    base_model.trainable = False

    inputs = keras.Input(shape=(150, 150, 3))

    base = base_model(inputs, training=False)
    vector = keras.layers.GlobalAveragePooling2D()(base)
    outputs = keras.layers.Dense(10)(vector)

    model = keras.Model(inputs, outputs)

    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate),
        loss=keras.losses.CategoricalCrossentropy(from_logits=True),
        metrics=["accuracy"],
    )

    return model
```

Watching metrics this way is not convenient, so let's create a special callback for that

In [ ]:

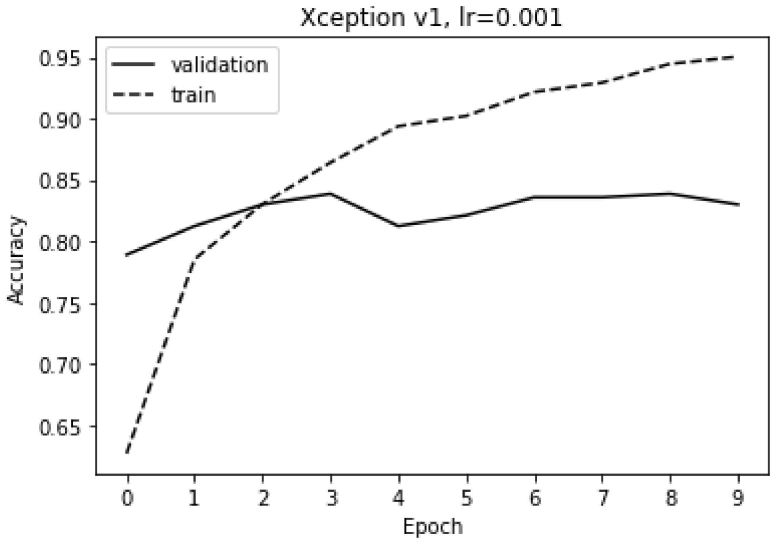
```
model = make_model(learning_rate=0.001)
history_0_001 = model.fit(train_ds, epochs=10, validation_data=val_ds)
```

```
Epoch 1/10
96/96 [==============================] - 17s 156ms/step - loss: 1.1042 - accuracy:
0.6268 - val_loss: 0.7033 - val_accuracy: 0.7889
Epoch 2/10
96/96 [==============================] - 14s 148ms/step - loss: 0.6387 - accuracy:
```

```
0.7852 - val_loss: 0.6148 - val_accuracy: 0.8123
Epoch 3/10
96/96 [==============================] - 14s 146ms/step - loss: 0.5095 - accuracy:
0.8302 - val_loss: 0.5685 - val_accuracy: 0.8299
Epoch 4/10
96/96 [==============================] - 15s 151ms/step - loss: 0.4277 - accuracy:
0.8641 - val_loss: 0.5522 - val_accuracy: 0.8387
Epoch 5/10
96/96 [==============================] - 14s 147ms/step - loss: 0.3711 - accuracy:
0.8937 - val_loss: 0.5682 - val_accuracy: 0.8123
Epoch 6/10
96/96 [==============================] - 14s 149ms/step - loss: 0.3370 - accuracy:
0.9022 - val_loss: 0.5497 - val_accuracy: 0.8211
Epoch 7/10
96/96 [==============================] - 14s 144ms/step - loss: 0.2904 - accuracy:
0.9218 - val_loss: 0.5262 - val_accuracy: 0.8358
Epoch 8/10
96/96 [==============================] - 14s 145ms/step - loss: 0.2608 - accuracy:
0.9293 - val_loss: 0.5250 - val_accuracy: 0.8358
Epoch 9/10
96/96 [==============================] - 15s 156ms/step - loss: 0.2336 - accuracy:
0.9446 - val_loss: 0.5190 - val_accuracy: 0.8387
Epoch 10/10
96/96 [==============================] - 14s 147ms/step - loss: 0.2108 - accuracy:
0.9505 - val_loss: 0.5299 - val_accuracy: 0.8299
```

In [ ]:

```
plt.figure(figsize=(6, 4))

epochs = history_0_001.epoch
val = history_0_001.history['val_accuracy']
train = history_0_001.history['accuracy']

plt.plot(epochs, val, color='black', linestyle='solid', label='validation')
plt.plot(epochs, train, color='black', linestyle='dashed', label='train')

plt.title('Xception v1, lr=0.001')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')

plt.xticks(epochs)

plt.legend()


plt.savefig('xception_v1_0_001.svg')

plt.show()
```

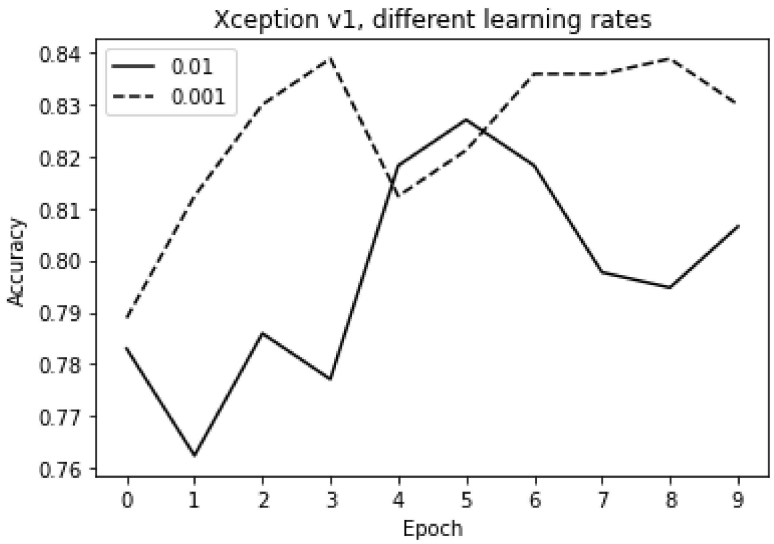


In [ ]:

```
epochs = np.arange(10)
val_0_01 = history.history['val_accuracy']
val_0_001 = history_0_001.history['val_accuracy']
```

In [ ]:

```
plt.figure(figsize=(6, 4))

plt.plot(epochs, val_0_01, color='black', linestyle='solid', label='0.01')
plt.plot(epochs, val_0_001, color='black', linestyle='dashed', label='0.001')


plt.title('Xception v1, different learning rates')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')

plt.xticks(epochs)

plt.legend()

plt.savefig('xception_v1_all_lr.svg')

plt.show()
```



The best models:

- learning rate 0.01: 0.8270

- learning rate 0.001: 0.8299

To save the best model, we can use a callback. It'll monitor the accuracy, and if it's an improvement over the previous version, it'll save the model to disk

In [ ]:
```
model = make_model(learning_rate=0.001)
```

In [ ]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        "xception_v1_{epoch:02d}_{val_accuracy:.3f}.h5",
        monitor="val_accuracy",
        save_best_only=True,
        mode='max'
    )
]

history_0_001 = model.fit(train_ds, epochs=10, validation_data=val_ds, callbacks=cal
```

```
Epoch 1/10
96/96 [==============================] - 16s 149ms/step - loss: 1.1204 - accuracy:
0.6193 - val_loss: 0.7065 - val_accuracy: 0.7889
/opt/conda/lib/python3.7/site-packages/keras/utils/generic_utils.py:497: CustomMaskW
arning: Custom mask layers require a config and must override get_config. When loadi
ng, the custom mask layer must be passed to the custom_objects argument.
  category=CustomMaskWarning)
Epoch 2/10
96/96 [==============================] - 14s 149ms/step - loss: 0.6256 - accuracy:
0.7904 - val_loss: 0.6526 - val_accuracy: 0.7918
Epoch 3/10
96/96 [==============================] - 14s 141ms/step - loss: 0.5105 - accuracy:
0.8312 - val_loss: 0.5938 - val_accuracy: 0.8123
Epoch 4/10
96/96 [==============================] - 15s 154ms/step - loss: 0.4288 - accuracy:
0.8703 - val_loss: 0.6086 - val_accuracy: 0.7889
Epoch 5/10
96/96 [==============================] - 14s 141ms/step - loss: 0.3766 - accuracy:
0.8804 - val_loss: 0.5578 - val_accuracy: 0.8211
Epoch 6/10
96/96 [==============================] - 14s 147ms/step - loss: 0.3257 - accuracy:
0.9051 - val_loss: 0.5587 - val_accuracy: 0.8182
Epoch 7/10
96/96 [==============================] - 14s 142ms/step - loss: 0.2948 - accuracy:
0.9179 - val_loss: 0.5433 - val_accuracy: 0.8270
Epoch 8/10
96/96 [==============================] - 15s 155ms/step - loss: 0.2562 - accuracy:
0.9351 - val_loss: 0.5487 - val_accuracy: 0.8270
Epoch 9/10
96/96 [==============================] - 14s 143ms/step - loss: 0.2332 - accuracy:
0.9387 - val_loss: 0.5495 - val_accuracy: 0.8240
Epoch 10/10
96/96 [==============================] - 15s 159ms/step - loss: 0.2139 - accuracy:
0.9472 - val_loss: 0.5589 - val_accuracy: 0.8152
```

Let's add one more layer - and a dropout between them

In [ ]:
```
def make_model(learning_rate, droprate):
    base_model = Xception(
        weights='imagenet',
        input_shape=(150, 150, 3),
        include_top=False
    )
```

```
    base_model.trainable = False

    inputs = keras.Input(shape=(150, 150, 3))

    base = base_model(inputs, training=False)
    vector = keras.layers.GlobalAveragePooling2D()(base)

    inner = keras.layers.Dense(100, activation='relu')(vector)
    drop = keras.layers.Dropout(droprate)(inner)

    outputs = keras.layers.Dense(10)(drop)

    model = keras.Model(inputs, outputs)

    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate),
        loss=keras.losses.CategoricalCrossentropy(from_logits=True),
        metrics=["accuracy"],
    )

    return model
```

In [ ]:

```
model = make_model(learning_rate=0.001, droprate=0.0)

callbacks = [
    keras.callbacks.ModelCheckpoint(
        "xception_v2_0_0_{epoch:02d}_{val_accuracy:.3f}.h5",
        monitor="val_accuracy",
        save_best_only=True,
        mode='max'
    )
]

history_0 = model.fit(train_ds, epochs=30, validation_data=val_ds, callbacks=callbac
```

```
Epoch 1/30
96/96 [==============================] - 18s 160ms/step - loss: 0.9495 - accuracy:
0.6754 - val_loss: 0.6415 - val_accuracy: 0.7918
Epoch 2/30
96/96 [==============================] - 14s 151ms/step - loss: 0.5050 - accuracy:
0.8292 - val_loss: 0.6105 - val_accuracy: 0.7742
Epoch 3/30
96/96 [==============================] - 14s 146ms/step - loss: 0.3512 - accuracy:
0.8758 - val_loss: 0.5341 - val_accuracy: 0.8240
Epoch 4/30
96/96 [==============================] - 16s 165ms/step - loss: 0.2453 - accuracy:
0.9241 - val_loss: 0.5628 - val_accuracy: 0.8152
Epoch 5/30
96/96 [==============================] - 14s 147ms/step - loss: 0.1723 - accuracy:
0.9518 - val_loss: 0.6080 - val_accuracy: 0.8006
Epoch 6/30
96/96 [==============================] - 18s 184ms/step - loss: 0.1224 - accuracy:
0.9729 - val_loss: 0.5782 - val_accuracy: 0.8123
Epoch 7/30
96/96 [==============================] - 18s 183ms/step - loss: 0.0850 - accuracy:
0.9873 - val_loss: 0.5849 - val_accuracy: 0.8270
Epoch 8/30
96/96 [==============================] - 16s 170ms/step - loss: 0.0556 - accuracy:
0.9964 - val_loss: 0.6034 - val_accuracy: 0.8299
Epoch 9/30
96/96 [==============================] - 14s 149ms/step - loss: 0.0383 - accuracy:
```

```
0.9977 - val_loss: 0.6156 - val_accuracy: 0.8182
Epoch 10/30
96/96 [==============================] - 15s 161ms/step - loss: 0.0299 - accuracy:
0.9987 - val_loss: 0.6221 - val_accuracy: 0.8358
Epoch 11/30
96/96 [==============================] - 15s 157ms/step - loss: 0.0222 - accuracy:
0.9997 - val_loss: 0.6471 - val_accuracy: 0.8299
Epoch 12/30
96/96 [==============================] - 17s 172ms/step - loss: 0.0199 - accuracy:
0.9993 - val_loss: 0.6764 - val_accuracy: 0.8358
Epoch 13/30
96/96 [==============================] - 14s 146ms/step - loss: 0.0142 - accuracy:
0.9997 - val_loss: 0.6822 - val_accuracy: 0.8123
Epoch 14/30
96/96 [==============================] - 15s 151ms/step - loss: 0.0152 - accuracy:
0.9993 - val_loss: 0.6728 - val_accuracy: 0.8211
Epoch 15/30
96/96 [==============================] - 15s 154ms/step - loss: 0.0126 - accuracy:
0.9993 - val_loss: 0.6976 - val_accuracy: 0.8270
Epoch 16/30
96/96 [==============================] - 14s 151ms/step - loss: 0.0123 - accuracy:
0.9993 - val_loss: 0.6941 - val_accuracy: 0.8299
Epoch 17/30
96/96 [==============================] - 15s 156ms/step - loss: 0.0081 - accuracy:
0.9997 - val_loss: 0.7125 - val_accuracy: 0.8270
Epoch 18/30
96/96 [==============================] - 15s 154ms/step - loss: 0.0095 - accuracy:
0.9993 - val_loss: 0.7098 - val_accuracy: 0.8182
Epoch 19/30
96/96 [==============================] - 16s 165ms/step - loss: 0.0069 - accuracy:
0.9997 - val_loss: 0.7366 - val_accuracy: 0.8240
Epoch 20/30
96/96 [==============================] - 14s 148ms/step - loss: 0.0091 - accuracy:
0.9990 - val_loss: 0.7304 - val_accuracy: 0.8270
Epoch 21/30
96/96 [==============================] - 16s 165ms/step - loss: 0.0067 - accuracy:
0.9993 - val_loss: 0.7367 - val_accuracy: 0.8299
Epoch 22/30
96/96 [==============================] - 14s 143ms/step - loss: 0.0052 - accuracy:
0.9997 - val_loss: 0.7482 - val_accuracy: 0.8299
Epoch 23/30
96/96 [==============================] - 15s 161ms/step - loss: 0.0094 - accuracy:
0.9990 - val_loss: 0.7738 - val_accuracy: 0.8211
Epoch 24/30
96/96 [==============================] - 14s 146ms/step - loss: 0.0084 - accuracy:
0.9993 - val_loss: 0.7770 - val_accuracy: 0.8240
Epoch 25/30
96/96 [==============================] - 14s 149ms/step - loss: 0.0061 - accuracy:
0.9993 - val_loss: 0.7820 - val_accuracy: 0.8240
Epoch 26/30
96/96 [==============================] - 14s 149ms/step - loss: 0.0102 - accuracy:
0.9990 - val_loss: 0.7856 - val_accuracy: 0.8152
Epoch 27/30
96/96 [==============================] - 15s 161ms/step - loss: 0.0067 - accuracy:
0.9984 - val_loss: 0.8432 - val_accuracy: 0.8094
Epoch 28/30
96/96 [==============================] - 14s 145ms/step - loss: 0.0059 - accuracy:
0.9990 - val_loss: 0.8530 - val_accuracy: 0.8123
Epoch 29/30
96/96 [==============================] - 15s 160ms/step - loss: 0.0526 - accuracy:
0.9817 - val_loss: 1.1074 - val_accuracy: 0.7625
Epoch 30/30
96/96 [==============================] - 14s 148ms/step - loss: 0.1854 - accuracy:
0.9384 - val_loss: 1.0493 - val_accuracy: 0.7713
```

In [ ]:

```
model = make_model(learning_rate=0.001, droprate=0.2)

callbacks = [
    keras.callbacks.ModelCheckpoint(
        "xception_v2_0_2_{epoch:02d}_{val_accuracy:.3f}.h5",
        monitor="val_accuracy",
        save_best_only=True,
        mode='max'
    )
]

history_1 = model.fit(train_ds, epochs=30, validation_data=val_ds, callbacks=callbac
```

```
Epoch 1/30
96/96 [==============================] - 17s 156ms/step - loss: 1.0716 - accuracy:
0.6372 - val_loss: 0.7098 - val_accuracy: 0.7889
Epoch 2/30
96/96 [==============================] - 16s 162ms/step - loss: 0.6193 - accuracy:
0.7826 - val_loss: 0.6023 - val_accuracy: 0.7918
Epoch 3/30
96/96 [==============================] - 14s 143ms/step - loss: 0.4581 - accuracy:
0.8442 - val_loss: 0.5410 - val_accuracy: 0.8211
Epoch 4/30
96/96 [==============================] - 15s 152ms/step - loss: 0.3575 - accuracy:
0.8814 - val_loss: 0.5806 - val_accuracy: 0.8035
Epoch 5/30
96/96 [==============================] - 15s 161ms/step - loss: 0.2847 - accuracy:
0.9045 - val_loss: 0.5507 - val_accuracy: 0.7947
Epoch 6/30
96/96 [==============================] - 14s 145ms/step - loss: 0.2278 - accuracy:
0.9286 - val_loss: 0.5552 - val_accuracy: 0.8123
Epoch 7/30
96/96 [==============================] - 15s 156ms/step - loss: 0.1825 - accuracy:
0.9465 - val_loss: 0.5396 - val_accuracy: 0.8182
Epoch 8/30
96/96 [==============================] - 14s 150ms/step - loss: 0.1319 - accuracy:
0.9638 - val_loss: 0.5775 - val_accuracy: 0.8152
Epoch 9/30
96/96 [==============================] - 15s 155ms/step - loss: 0.1125 - accuracy:
0.9710 - val_loss: 0.6285 - val_accuracy: 0.7977
Epoch 10/30
96/96 [==============================] - 14s 147ms/step - loss: 0.0978 - accuracy:
0.9782 - val_loss: 0.6006 - val_accuracy: 0.8065
Epoch 11/30
96/96 [==============================] - 16s 164ms/step - loss: 0.0818 - accuracy:
0.9788 - val_loss: 0.6247 - val_accuracy: 0.8006
Epoch 12/30
96/96 [==============================] - 14s 143ms/step - loss: 0.0729 - accuracy:
0.9827 - val_loss: 0.6396 - val_accuracy: 0.8211
Epoch 13/30
96/96 [==============================] - 16s 164ms/step - loss: 0.0556 - accuracy:
0.9899 - val_loss: 0.6140 - val_accuracy: 0.8328
Epoch 14/30
96/96 [==============================] - 14s 144ms/step - loss: 0.0534 - accuracy:
0.9905 - val_loss: 0.6190 - val_accuracy: 0.8270
Epoch 15/30
96/96 [==============================] - 14s 150ms/step - loss: 0.0469 - accuracy:
0.9909 - val_loss: 0.6705 - val_accuracy: 0.8152
Epoch 16/30
96/96 [==============================] - 16s 163ms/step - loss: 0.0413 - accuracy:
0.9932 - val_loss: 0.6848 - val_accuracy: 0.8123
Epoch 17/30
96/96 [==============================] - 14s 148ms/step - loss: 0.0370 - accuracy:
```

```
0.9932 - val_loss: 0.6987 - val_accuracy: 0.8211
Epoch 18/30
96/96 [==============================] - 15s 161ms/step - loss: 0.0291 - accuracy:
0.9948 - val_loss: 0.6575 - val_accuracy: 0.8475
Epoch 19/30
96/96 [==============================] - 15s 151ms/step - loss: 0.0223 - accuracy:
0.9974 - val_loss: 0.6810 - val_accuracy: 0.8211
Epoch 20/30
96/96 [==============================] - 15s 157ms/step - loss: 0.0224 - accuracy:
0.9980 - val_loss: 0.7337 - val_accuracy: 0.8065
Epoch 21/30
96/96 [==============================] - 14s 141ms/step - loss: 0.0307 - accuracy:
0.9938 - val_loss: 0.7286 - val_accuracy: 0.8065
Epoch 22/30
96/96 [==============================] - 14s 145ms/step - loss: 0.0255 - accuracy:
0.9954 - val_loss: 0.7651 - val_accuracy: 0.8152
Epoch 23/30
96/96 [==============================] - 14s 143ms/step - loss: 0.0278 - accuracy:
0.9951 - val_loss: 0.7397 - val_accuracy: 0.8065
Epoch 24/30
96/96 [==============================] - 15s 158ms/step - loss: 0.0247 - accuracy:
0.9941 - val_loss: 0.7901 - val_accuracy: 0.8035
Epoch 25/30
96/96 [==============================] - 14s 141ms/step - loss: 0.0221 - accuracy:
0.9938 - val_loss: 0.7749 - val_accuracy: 0.8152
Epoch 26/30
96/96 [==============================] - 16s 163ms/step - loss: 0.0177 - accuracy:
0.9967 - val_loss: 0.7291 - val_accuracy: 0.8328
Epoch 27/30
96/96 [==============================] - 14s 144ms/step - loss: 0.0224 - accuracy:
0.9954 - val_loss: 0.7527 - val_accuracy: 0.8211
Epoch 28/30
96/96 [==============================] - 14s 145ms/step - loss: 0.0125 - accuracy:
0.9984 - val_loss: 0.7677 - val_accuracy: 0.8182
Epoch 29/30
96/96 [==============================] - 15s 158ms/step - loss: 0.0136 - accuracy:
0.9984 - val_loss: 0.8081 - val_accuracy: 0.8094
Epoch 30/30
96/96 [==============================] - 14s 144ms/step - loss: 0.0233 - accuracy:
0.9919 - val_loss: 0.8597 - val_accuracy: 0.7947
```

In [ ]:

```
model = make_model(learning_rate=0.001, droprate=0.5)

callbacks = [
    keras.callbacks.ModelCheckpoint(
        "xception_v2_0_5_{epoch:02d}_{val_accuracy:.3f}.h5",
        monitor="val_accuracy",
        save_best_only=True,
        mode='max'
    )
]

history_2 = model.fit(train_ds, epochs=30, validation_data=val_ds, callbacks=callbac
```

```
Epoch 1/30
96/96 [==============================] - 17s 156ms/step - loss: 1.3059 - accuracy:
0.5671 - val_loss: 0.7618 - val_accuracy: 0.7507
Epoch 2/30
96/96 [==============================] - 21s 215ms/step - loss: 0.8471 - accuracy:
0.7115 - val_loss: 0.6414 - val_accuracy: 0.7889
Epoch 3/30
96/96 [==============================] - 20s 211ms/step - loss: 0.6992 - accuracy:
0.7686 - val_loss: 0.6063 - val_accuracy: 0.7977
```

```
Epoch 4/30
96/96 [==============================] - 20s 205ms/step - loss: 0.6054 - accuracy:
0.7819 - val_loss: 0.6021 - val_accuracy: 0.8035
Epoch 5/30
96/96 [==============================] - 18s 184ms/step - loss: 0.5320 - accuracy:
0.8194 - val_loss: 0.5613 - val_accuracy: 0.8094
Epoch 6/30
96/96 [==============================] - 16s 161ms/step - loss: 0.4443 - accuracy:
0.8406 - val_loss: 0.5596 - val_accuracy: 0.8006
Epoch 7/30
96/96 [==============================] - 14s 146ms/step - loss: 0.4188 - accuracy:
0.8517 - val_loss: 0.5262 - val_accuracy: 0.8182
Epoch 8/30
96/96 [==============================] - 16s 163ms/step - loss: 0.3693 - accuracy:
0.8696 - val_loss: 0.5396 - val_accuracy: 0.8006
Epoch 9/30
96/96 [==============================] - 14s 149ms/step - loss: 0.3156 - accuracy:
0.8911 - val_loss: 0.6042 - val_accuracy: 0.8035
Epoch 10/30
96/96 [==============================] - 16s 162ms/step - loss: 0.2963 - accuracy:
0.9055 - val_loss: 0.5500 - val_accuracy: 0.8065
Epoch 11/30
96/96 [==============================] - 15s 152ms/step - loss: 0.2717 - accuracy:
0.9094 - val_loss: 0.5611 - val_accuracy: 0.7918
Epoch 12/30
96/96 [==============================] - 16s 164ms/step - loss: 0.2461 - accuracy:
0.9156 - val_loss: 0.5673 - val_accuracy: 0.8094
Epoch 13/30
96/96 [==============================] - 15s 153ms/step - loss: 0.2468 - accuracy:
0.9166 - val_loss: 0.5243 - val_accuracy: 0.8270
Epoch 14/30
96/96 [==============================] - 16s 165ms/step - loss: 0.2048 - accuracy:
0.9299 - val_loss: 0.5651 - val_accuracy: 0.8211
Epoch 15/30
96/96 [==============================] - 15s 156ms/step - loss: 0.1920 - accuracy:
0.9342 - val_loss: 0.5433 - val_accuracy: 0.8152
Epoch 16/30
96/96 [==============================] - 14s 148ms/step - loss: 0.1789 - accuracy:
0.9446 - val_loss: 0.5600 - val_accuracy: 0.8123
Epoch 17/30
96/96 [==============================] - 17s 173ms/step - loss: 0.1641 - accuracy:
0.9446 - val_loss: 0.6261 - val_accuracy: 0.8270
Epoch 18/30
96/96 [==============================] - 15s 150ms/step - loss: 0.1489 - accuracy:
0.9540 - val_loss: 0.5933 - val_accuracy: 0.8270
Epoch 19/30
96/96 [==============================] - 16s 168ms/step - loss: 0.1508 - accuracy:
0.9508 - val_loss: 0.6073 - val_accuracy: 0.8328
Epoch 20/30
96/96 [==============================] - 15s 154ms/step - loss: 0.1375 - accuracy:
0.9537 - val_loss: 0.6173 - val_accuracy: 0.8387
Epoch 21/30
96/96 [==============================] - 16s 167ms/step - loss: 0.1310 - accuracy:
0.9596 - val_loss: 0.6268 - val_accuracy: 0.8211
Epoch 22/30
96/96 [==============================] - 15s 154ms/step - loss: 0.1165 - accuracy:
0.9661 - val_loss: 0.6740 - val_accuracy: 0.8123
Epoch 23/30
96/96 [==============================] - 14s 145ms/step - loss: 0.1141 - accuracy:
0.9619 - val_loss: 0.6535 - val_accuracy: 0.8328
Epoch 24/30
96/96 [==============================] - 16s 168ms/step - loss: 0.1142 - accuracy:
0.9632 - val_loss: 0.6283 - val_accuracy: 0.8270
Epoch 25/30
```

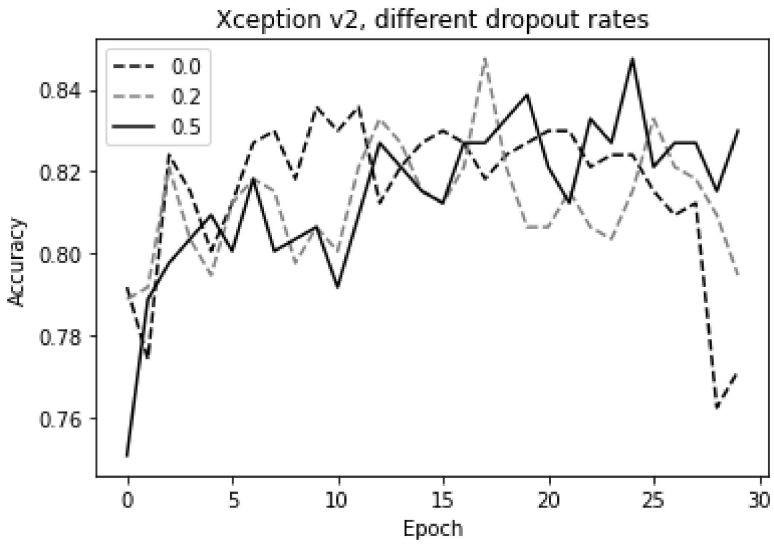
```
96/96 [==============================] - 14s 147ms/step - loss: 0.1100 - accuracy:
0.9648 - val_loss: 0.6352 - val_accuracy: 0.8475
Epoch 26/30
96/96 [==============================] - 16s 169ms/step - loss: 0.1042 - accuracy:
0.9684 - val_loss: 0.7045 - val_accuracy: 0.8211
Epoch 27/30
96/96 [==============================] - 14s 143ms/step - loss: 0.0921 - accuracy:
0.9713 - val_loss: 0.7170 - val_accuracy: 0.8270
Epoch 28/30
96/96 [==============================] - 16s 170ms/step - loss: 0.0998 - accuracy:
0.9684 - val_loss: 0.6587 - val_accuracy: 0.8270
Epoch 29/30
96/96 [==============================] - 14s 147ms/step - loss: 0.0989 - accuracy:
0.9668 - val_loss: 0.6957 - val_accuracy: 0.8152
Epoch 30/30
96/96 [==============================] - 14s 143ms/step - loss: 0.0862 - accuracy:
0.9739 - val_loss: 0.7273 - val_accuracy: 0.8299
```

In [ ]:

```
epochs = history_0.epoch

train00 = history_0.history['accuracy']
train02 = history_1.history['accuracy']
train05 = history_2.history['accuracy']

val00 = history_0.history['val_accuracy']
val02 = history_1.history['val_accuracy']
val05 = history_2.history['val_accuracy']
```

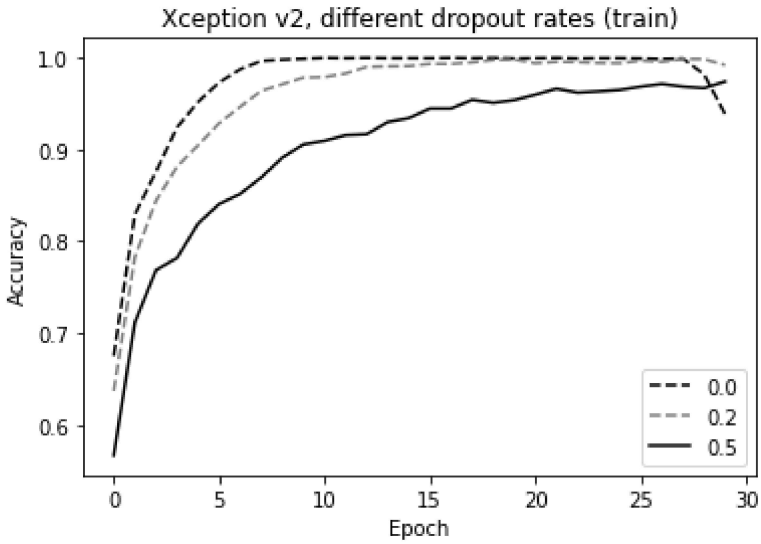
In [ ]:

```
plt.figure(figsize=(6, 4))

plt.plot(epochs, val00, color='black', linestyle='dashed', label='0.0')
plt.plot(epochs, val02, color='grey', linestyle='dashed', label='0.2')
plt.plot(epochs, val05, color='black', linestyle='solid', label='0.5')

plt.title('Xception v2, different dropout rates')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')


plt.legend()

plt.savefig('xception_v2_dropout.svg')

plt.show()
```



In [ ]:

```
plt.figure(figsize=(6, 4))

plt.plot(epochs, train00, color='black', linestyle='dashed', label='0.0')
plt.plot(epochs, train02, color='grey', linestyle='dashed', label='0.2')
plt.plot(epochs, train05, color='black', linestyle='solid', label='0.5')

plt.title('Xception v2, different dropout rates (train)')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')


plt.legend()

plt.savefig('xception_v2_dropout_train.svg')

plt.show()
```



Data augmentation

In [ ]:

```
validation_gen = ImageDataGenerator(preprocessing_function=preprocess_input)

val_ds = validation_gen.flow_from_directory(
    "../input/clothing-dataset-small/validation",
    seed=1,
    target_size=image_size,
    batch_size=batch_size,
)
```

```
Found 341 images belonging to 10 classes.
```

In [ ]:

```
train_gen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    shear_range=10.0,
    zoom_range=0.1,
    horizontal_flip=True,
)

train_ds = train_gen.flow_from_directory(
    "../input/clothing-dataset-small/train",
    seed=1,
    target_size=image_size,
```

```
    batch_size=batch_size,
)
```

```
Found 3068 images belonging to 10 classes.
```

In [ ]:

```
model = make_model(learning_rate=0.001, droprate=0.2)

callbacks = [
    keras.callbacks.ModelCheckpoint(
        "xception_v3_{epoch:02d}_{val_accuracy:.3f}.h5",
        monitor="val_accuracy",
        save_best_only=True,
        mode='max'
    )
]

history = model.fit(train_ds, epochs=50, validation_data=val_ds, callbacks=callbacks
```

```
Epoch 1/50
96/96 [==============================] - 31s 297ms/step - loss: 1.1287 - accuracy:
0.6154 - val_loss: 0.7393 - val_accuracy: 0.7478
Epoch 2/50
96/96 [==============================] - 29s 303ms/step - loss: 0.7346 - accuracy:
0.7552 - val_loss: 0.6713 - val_accuracy: 0.7889
Epoch 3/50
96/96 [==============================] - 30s 312ms/step - loss: 0.5960 - accuracy:
0.7976 - val_loss: 0.5795 - val_accuracy: 0.8211
Epoch 4/50
96/96 [==============================] - 30s 312ms/step - loss: 0.5375 - accuracy:
0.8178 - val_loss: 0.5716 - val_accuracy: 0.8035
Epoch 5/50
96/96 [==============================] - 30s 313ms/step - loss: 0.4618 - accuracy:
0.8360 - val_loss: 0.6189 - val_accuracy: 0.7830
Epoch 6/50
96/96 [==============================] - 30s 309ms/step - loss: 0.4089 - accuracy:
0.8592 - val_loss: 0.5806 - val_accuracy: 0.8123
Epoch 7/50
96/96 [==============================] - 30s 308ms/step - loss: 0.3786 - accuracy:
0.8716 - val_loss: 0.6073 - val_accuracy: 0.8035
Epoch 8/50
96/96 [==============================] - 27s 277ms/step - loss: 0.3571 - accuracy:
0.8758 - val_loss: 0.5992 - val_accuracy: 0.8035
Epoch 9/50
96/96 [==============================] - 30s 317ms/step - loss: 0.3040 - accuracy:
0.8947 - val_loss: 0.5841 - val_accuracy: 0.8006
Epoch 10/50
96/96 [==============================] - 29s 307ms/step - loss: 0.2999 - accuracy:
0.8944 - val_loss: 0.6204 - val_accuracy: 0.8094
Epoch 11/50
96/96 [==============================] - 30s 316ms/step - loss: 0.2801 - accuracy:
0.9025 - val_loss: 0.5875 - val_accuracy: 0.8240
Epoch 12/50
96/96 [==============================] - 29s 299ms/step - loss: 0.2616 - accuracy:
0.9120 - val_loss: 0.5688 - val_accuracy: 0.8299
Epoch 13/50
96/96 [==============================] - 30s 317ms/step - loss: 0.2316 - accuracy:
0.9218 - val_loss: 0.5340 - val_accuracy: 0.8182
Epoch 14/50
96/96 [==============================] - 28s 296ms/step - loss: 0.2347 - accuracy:
0.9175 - val_loss: 0.5355 - val_accuracy: 0.8328
Epoch 15/50
96/96 [==============================] - 30s 313ms/step - loss: 0.2153 - accuracy:
0.9263 - val_loss: 0.5816 - val_accuracy: 0.8328
```

```
Epoch 16/50
96/96 [==============================] - 27s 282ms/step - loss: 0.2084 - accuracy:
0.9309 - val_loss: 0.5819 - val_accuracy: 0.8270
Epoch 17/50
96/96 [==============================] - 30s 308ms/step - loss: 0.2177 - accuracy:
0.9319 - val_loss: 0.5816 - val_accuracy: 0.8182
Epoch 18/50
96/96 [==============================] - 30s 317ms/step - loss: 0.1817 - accuracy:
0.9374 - val_loss: 0.5788 - val_accuracy: 0.8270
Epoch 19/50
96/96 [==============================] - 31s 318ms/step - loss: 0.1829 - accuracy:
0.9368 - val_loss: 0.6485 - val_accuracy: 0.8094
Epoch 20/50
96/96 [==============================] - 30s 315ms/step - loss: 0.1691 - accuracy:
0.9426 - val_loss: 0.5969 - val_accuracy: 0.8387
Epoch 21/50
96/96 [==============================] - 28s 293ms/step - loss: 0.1737 - accuracy:
0.9400 - val_loss: 0.6138 - val_accuracy: 0.8152
Epoch 22/50
96/96 [==============================] - 30s 309ms/step - loss: 0.1444 - accuracy:
0.9544 - val_loss: 0.6129 - val_accuracy: 0.8387
Epoch 23/50
96/96 [==============================] - 27s 276ms/step - loss: 0.1459 - accuracy:
0.9498 - val_loss: 0.6547 - val_accuracy: 0.8299
Epoch 24/50
96/96 [==============================] - 30s 308ms/step - loss: 0.1236 - accuracy:
0.9583 - val_loss: 0.6621 - val_accuracy: 0.8358
Epoch 25/50
96/96 [==============================] - 31s 318ms/step - loss: 0.1325 - accuracy:
0.9557 - val_loss: 0.6104 - val_accuracy: 0.8387
Epoch 26/50
96/96 [==============================] - 30s 317ms/step - loss: 0.1424 - accuracy:
0.9511 - val_loss: 0.6272 - val_accuracy: 0.8328
Epoch 27/50
96/96 [==============================] - 30s 317ms/step - loss: 0.1470 - accuracy:
0.9534 - val_loss: 0.6846 - val_accuracy: 0.8123
Epoch 28/50
96/96 [==============================] - 27s 281ms/step - loss: 0.1220 - accuracy:
0.9580 - val_loss: 0.6203 - val_accuracy: 0.8446
Epoch 29/50
96/96 [==============================] - 27s 281ms/step - loss: 0.1104 - accuracy:
0.9654 - val_loss: 0.6785 - val_accuracy: 0.8123
Epoch 30/50
96/96 [==============================] - 31s 319ms/step - loss: 0.1263 - accuracy:
0.9544 - val_loss: 0.6821 - val_accuracy: 0.8328
Epoch 31/50
96/96 [==============================] - 28s 289ms/step - loss: 0.1060 - accuracy:
0.9635 - val_loss: 0.6748 - val_accuracy: 0.8123
Epoch 32/50
96/96 [==============================] - 33s 344ms/step - loss: 0.1117 - accuracy:
0.9638 - val_loss: 0.7213 - val_accuracy: 0.8240
Epoch 33/50
96/96 [==============================] - 30s 312ms/step - loss: 0.1062 - accuracy:
0.9622 - val_loss: 0.7883 - val_accuracy: 0.8152
Epoch 34/50
96/96 [==============================] - 34s 355ms/step - loss: 0.1151 - accuracy:
0.9586 - val_loss: 0.6919 - val_accuracy: 0.8299
Epoch 35/50
96/96 [==============================] - 31s 320ms/step - loss: 0.1255 - accuracy:
0.9570 - val_loss: 0.7025 - val_accuracy: 0.8123
Epoch 36/50
96/96 [==============================] - 30s 316ms/step - loss: 0.1130 - accuracy:
0.9593 - val_loss: 0.6849 - val_accuracy: 0.8182
Epoch 37/50
```

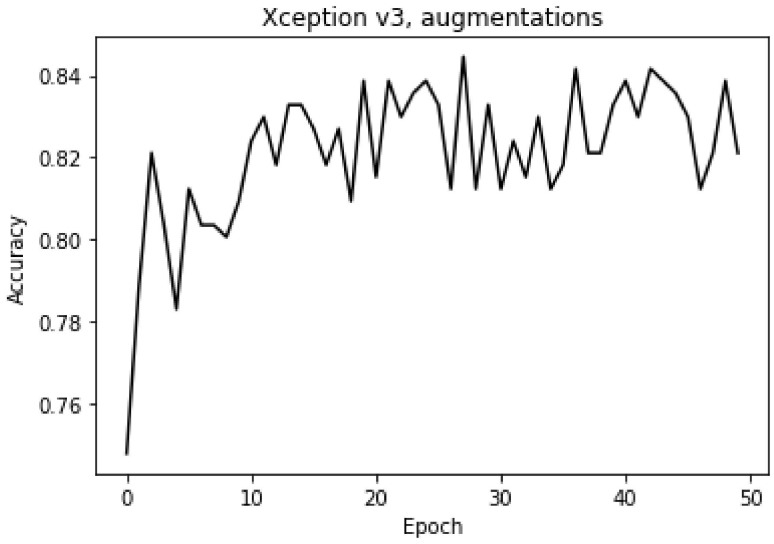
```
96/96 [==============================] - 27s 285ms/step - loss: 0.1187 - accuracy:
0.9576 - val_loss: 0.6796 - val_accuracy: 0.8416
Epoch 38/50
96/96 [==============================] - 28s 286ms/step - loss: 0.0921 - accuracy:
0.9674 - val_loss: 0.6874 - val_accuracy: 0.8211
Epoch 39/50
96/96 [==============================] - 32s 334ms/step - loss: 0.1016 - accuracy:
0.9694 - val_loss: 0.6986 - val_accuracy: 0.8211
Epoch 40/50
96/96 [==============================] - 29s 298ms/step - loss: 0.1016 - accuracy:
0.9654 - val_loss: 0.6937 - val_accuracy: 0.8328
Epoch 41/50
96/96 [==============================] - 31s 324ms/step - loss: 0.0958 - accuracy:
0.9671 - val_loss: 0.7439 - val_accuracy: 0.8387
Epoch 42/50
96/96 [==============================] - 28s 292ms/step - loss: 0.1024 - accuracy:
0.9645 - val_loss: 0.6942 - val_accuracy: 0.8299
Epoch 43/50
96/96 [==============================] - 31s 327ms/step - loss: 0.0866 - accuracy:
0.9733 - val_loss: 0.8148 - val_accuracy: 0.8416
Epoch 44/50
96/96 [==============================] - 28s 290ms/step - loss: 0.0728 - accuracy:
0.9756 - val_loss: 0.7707 - val_accuracy: 0.8387
Epoch 45/50
96/96 [==============================] - 28s 293ms/step - loss: 0.0823 - accuracy:
0.9726 - val_loss: 0.7566 - val_accuracy: 0.8358
Epoch 46/50
96/96 [==============================] - 32s 327ms/step - loss: 0.0798 - accuracy:
0.9729 - val_loss: 0.7843 - val_accuracy: 0.8299
Epoch 47/50
96/96 [==============================] - 27s 286ms/step - loss: 0.0818 - accuracy:
0.9707 - val_loss: 0.8318 - val_accuracy: 0.8123
Epoch 48/50
96/96 [==============================] - 31s 321ms/step - loss: 0.0767 - accuracy:
0.9733 - val_loss: 0.8636 - val_accuracy: 0.8211
Epoch 49/50
96/96 [==============================] - 27s 278ms/step - loss: 0.0844 - accuracy:
0.9720 - val_loss: 0.8295 - val_accuracy: 0.8387
Epoch 50/50
96/96 [==============================] - 31s 323ms/step - loss: 0.0846 - accuracy:
0.9723 - val_loss: 0.8201 - val_accuracy: 0.8211
```

In [ ]:

```
epochs = history.epoch
accuracy = history.history['val_accuracy']
```

In [ ]:

```
plt.figure(figsize=(6, 4))

plt.plot(epochs, accuracy, color='black', linestyle='solid')


plt.title('Xception v3, augmentations')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')

plt.savefig('xception_v3_aug.svg')

plt.show()
```



Larger model

In [ ]:

```
def make_model(learning_rate, droprate):
    base_model = Xception(
        weights='imagenet',
        input_shape=(299, 299, 3),
        include_top=False
    )

    base_model.trainable = False

    inputs = keras.Input(shape=(299, 299, 3))

    base = base_model(inputs, training=False)
    vector = keras.layers.GlobalAveragePooling2D()(base)

    inner = keras.layers.Dense(100, activation='relu')(vector)
    drop = keras.layers.Dropout(droprate)(inner)

    outputs = keras.layers.Dense(10)(drop)

    model = keras.Model(inputs, outputs)

    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate),
        loss=keras.losses.CategoricalCrossentropy(from_logits=True),
        metrics=["accuracy"],
    )

    return model
```

In [ ]:

```
image_size = (299, 299)
batch_size = 32
```

In [ ]:

```
train_gen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    shear_range=10.0,
    zoom_range=0.1,
    horizontal_flip=True,
)
```

```
train_ds = train_gen.flow_from_directory(
    "../input/clothing-dataset-small/train",
    seed=1,
    target_size=image_size,
    batch_size=batch_size,
)
```

```
Found 3068 images belonging to 10 classes.
```

In [ ]:

```
validation_gen = ImageDataGenerator(preprocessing_function=preprocess_input)

val_ds = validation_gen.flow_from_directory(
    "../input/clothing-dataset-small/validation",
    seed=1,
    target_size=image_size,
    batch_size=batch_size,
)
```

```
Found 341 images belonging to 10 classes.
```

In [ ]:

```
model = make_model(learning_rate=0.001, droprate=0.2)

callbacks = [
    keras.callbacks.ModelCheckpoint(
        "xception_v4_large_{epoch:02d}_{val_accuracy:.3f}.h5",
        monitor="val_accuracy",
        save_best_only=True,
        mode='max'
    )
]

history_l = model.fit(train_ds, epochs=20, validation_data=val_ds, callbacks=callbac
```

```
Epoch 1/20
96/96 [==============================] - 82s 827ms/step - loss: 0.8140 - accuracy:
0.7373 - val_loss: 0.4638 - val_accuracy: 0.8563
Epoch 2/20
96/96 [==============================] - 79s 824ms/step - loss: 0.4648 - accuracy:
0.8458 - val_loss: 0.4288 - val_accuracy: 0.8446
Epoch 3/20
96/96 [==============================] - 78s 812ms/step - loss: 0.3850 - accuracy:
0.8677 - val_loss: 0.3788 - val_accuracy: 0.8680
Epoch 4/20
96/96 [==============================] - 78s 808ms/step - loss: 0.3450 - accuracy:
0.8814 - val_loss: 0.4059 - val_accuracy: 0.8592
Epoch 5/20
96/96 [==============================] - 78s 814ms/step - loss: 0.3038 - accuracy:
0.8960 - val_loss: 0.4335 - val_accuracy: 0.8592
Epoch 6/20
96/96 [==============================] - 78s 815ms/step - loss: 0.2578 - accuracy:
0.9123 - val_loss: 0.4019 - val_accuracy: 0.8739
Epoch 7/20
96/96 [==============================] - 79s 822ms/step - loss: 0.2416 - accuracy:
0.9113 - val_loss: 0.3851 - val_accuracy: 0.8827
Epoch 8/20
96/96 [==============================] - 78s 808ms/step - loss: 0.2212 - accuracy:
0.9221 - val_loss: 0.4126 - val_accuracy: 0.8710
Epoch 9/20
96/96 [==============================] - 77s 802ms/step - loss: 0.2082 - accuracy:
0.9276 - val_loss: 0.3672 - val_accuracy: 0.8798
Epoch 10/20
96/96 [==============================] - 78s 812ms/step - loss: 0.1852 - accuracy:
```

```
0.9374 - val_loss: 0.3997 - val_accuracy: 0.8680
Epoch 11/20
96/96 [==============================] - 77s 799ms/step - loss: 0.1844 - accuracy:
0.9361 - val_loss: 0.3908 - val_accuracy: 0.8798
Epoch 12/20
96/96 [==============================] - 76s 796ms/step - loss: 0.1785 - accuracy:
0.9390 - val_loss: 0.4071 - val_accuracy: 0.8534
Epoch 13/20
96/96 [==============================] - 78s 815ms/step - loss: 0.1593 - accuracy:
0.9475 - val_loss: 0.3874 - val_accuracy: 0.8592
Epoch 14/20
96/96 [==============================] - 73s 765ms/step - loss: 0.1491 - accuracy:
0.9469 - val_loss: 0.4024 - val_accuracy: 0.8622
Epoch 15/20
96/96 [==============================] - 78s 808ms/step - loss: 0.1262 - accuracy:
0.9576 - val_loss: 0.3813 - val_accuracy: 0.8798
Epoch 16/20
96/96 [==============================] - 76s 789ms/step - loss: 0.1422 - accuracy:
0.9478 - val_loss: 0.3811 - val_accuracy: 0.8798
Epoch 17/20
96/96 [==============================] - 76s 793ms/step - loss: 0.1251 - accuracy:
0.9570 - val_loss: 0.3974 - val_accuracy: 0.8768
Epoch 18/20
96/96 [==============================] - 76s 796ms/step - loss: 0.1040 - accuracy:
0.9668 - val_loss: 0.3973 - val_accuracy: 0.8768
Epoch 19/20
96/96 [==============================] - 76s 789ms/step - loss: 0.1044 - accuracy:
0.9668 - val_loss: 0.4473 - val_accuracy: 0.8768
Epoch 20/20
96/96 [==============================] - 76s 794ms/step - loss: 0.1035 - accuracy:
0.9625 - val_loss: 0.4605 - val_accuracy: 0.8563
```

Let's test these models

In [ ]:

```
labels = {
    0: 'dress',
    1: 'hat',
    2: 'longsleeve',
    3: 'outwear',
    4: 'pants',
    5: 'shirt',
    6: 'shoes',
    7: 'shorts',
    8: 'skirt',
    9: 't-shirt'
}
```

Big model

In [ ]:

```
image_size = (299, 299)
```

In [ ]:

```
#for load pretrained model
# model = keras.models.load_model('./xception_v4_large_07_0.883.h5')
```

In [ ]:

```
path = '../input/clothing-dataset-small/train/pants/03b5fa92-c65d-4b45-820b-967e85f4
img = load_img(path, target_size=(image_size))
img
```

Out[ ]:

In [ ]:

```
x = np.array(img)
X = np.array([x])
X = preprocess_input(X)
```

In [ ]:

```
pred = model.predict(X)
labels[pred[0].argmax()]
```

Out[ ]:

```
'pants'
```