

# thesis\_main\_v9

August 16, 2021

## 1 The implementation of thesis project with keras/tensorflow

### 2 Introduction

In this Python notesbook, we choose the following as baselines and evaluate their performances: - Supervised learning with simple convolutional network (training from scratch) - Supervised learning with MLP (training from scratch) - Supervised learning with ResNet50 - Supervised learning with VGG16 - Supervised learning with MobileNet

We then evaluate our self-supervised contrastive model, which uses InfoNCE, known as NT-Xent (Normalized Temperature-scaled Cross Entropy Loss) in SimCLR and used in other works like MoCo.

The goal of our texture classification model is to train an embedding space constrained by contrastive loss, our model consists of two image augmentation transformers, base encoder, a projection head, and a pure linear regressor with cross-entropy loss as the customised accuracy metric.

We evaluate our proposed model with: - Contrastive learning with simple convolutional network as encoder (training from scratch) - Contrastive learning with VGG16 as encoder - Contrastive learning with Vision Transformer (ViT) as encoder (training from scratch)

### 3 Importing libraries

```
[1]: import tensorflow as tf
import tensorflow_addons as tfa
from tensorflow import keras

import numpy as np
import pandas as pd
import os
import sys
import random

import matplotlib.pyplot as plt

# enable GPU on macOS
#import os
#os.environ["KERAS_BACKEND"] = "plaidml.keras.backend"
```

```
#os.environ["RUNFILES_DIR"] = "/usr/local/anaconda3/share/plaidml"
#os.environ["PLAIDML_NATIVE_PATH"] = "/usr/local/anaconda3/lib/libplaidml.dylib"
#import keras
```

## 4 Setting local library path

Our modules lie in the directory *lib*, we add the path to the library searching path:

```
[2]: import numpy as np
import pandas as pd
import os
import sys

#libpath = "lib"
#sys.path.append(libpath)
```

## 5 Configurations

```
[3]: #image_size = (128, 128)
image_size = (64, 64) #faster for running on laptop

#dataset_root_orig = ""

n_image_channels = 3

#input image shape
image_shape = (*image_size, n_image_channels)

# sample batch size
batch_size = 128
#batch_size = 64

# how many epochs
n_epochs = 20

temperature = 0.2
#temperature = 0.5

#relatively slight augmentation
imgaug_params = {
    "min_area": 0.75,
    "brightness": 0.3,
```

```

        "jitter": 0.1
    }

#relatively strong augmentation


```

## 6 Bootstrapping the dataset

### 6.1 Image augmentation model for dataset bootstrapping

```
[345]: from tensorflow.keras.layers.experimental import preprocessing as u
         keras_preprocessing_layers
```

```

class CustomisedBootstrapImageAugmenter(keras.layers.Layer):
    def __init__(self,
                 target_size = (128, 128),
                 seed = 1121,
                 name = 'bootstrap_imgaug_layer',
                 **kwargs):

        self.target_size = target_size
        self.seed = seed

        self.model = keras.Sequential([
            #keras_preprocessing_layers.Rescaling(1.0 / 255),
            #keras_preprocessing_layers.Normalization(),
            keras_preprocessing_layers.RandomFlip(
                "horizontal",
                seed = self.seed
            ),
            keras_preprocessing_layers.RandomZoom(
                height_factor=0.3,
                width_factor=0.3,
                seed = self.seed
            ),
            keras_preprocessing_layers.RandomRotation(
                0.5,
                seed = self.seed
            ),
            keras_preprocessing_layers.RandomCrop(
                *self.target_size,
                seed = self.seed

```

```

        )

    ] ,
    name = "bootstrap_imgaug_model")

    super(CustomisedBootstrapImageAugmenter, self).__init__(name = name, **kwargs)

def get_config(self):
    config = super().get_config()

    config['target_size'] = self.target_size
    config['seed'] = self.seed
    config['model'] = self.model

    return config

def call(self, images, training = True):
    if training:
        images = self.model(images)

    return images

@classmethod
def from_config(cls, config):
    return cls(**config)

```

## 7 The facilities of operating the original dataset

```
[19]: import numpy as np
import pandas as pd
import os
import sys

TEXTURE_DS_IMAGE_SIZE = (512, 512)

def get_texture_subdirs():
    dataset_root = "datasets_orig"

    subdirs = [d.path for d in os.scandir(dataset_root)
              if d.is_dir(follow_symlinks = False) and
              not d.name.startswith(".") and
              not d.name.endswith("_npy")]

    subdirs = [s.rsplit("/", 1)[1] for s in subdirs]
```

```

    return subdirs

[20]: get_texture_subdirs()

[20]: ['Normalised_Brodatz_Texture',
        'Multiband_Brodatz_Texture',
        'Colored_Brodatz_Texture']

[23]: def load_texture_dataset_xy(*, image_size = TEXTURE_DS_IMAGE_SIZE,
                                dataset_name):

    assert dataset_name is not None, "The dataset_name must be specified."

    #dataset_path = "datasets_orig" + os.path.sep + "Multiband_Brodatz_Texture"
    dataset_path = "datasets_orig" + os.path.sep + dataset_name

    assert os.path.exists(dataset_path), f"{dataset_path} does not exist."


    print()
    print(f"Loading dataset from path: {dataset_path} ...")
    print("image size: ", image_size)

    X = []
    y = []

    # Get the subdirs for given dataset path

    files = []

    suffixes = (".tif", ".gif", ".png", ".jpg")

    imgfiles = [d.path for d in os.scandir(dataset_path)
                if d.is_file(follow_symlinks = False) and
                   not d.name.startswith(".") and
                   d.name.lower().endswith(suffixes) ]

    #print(imgfiles)

    for p in sorted(imgfiles):
        #print(p)

        # Loading images and pre-processing images
        img = keras.preprocessing.image.load_img( path = p,

```

```

        grayscale = False,
        color_mode='rgb',
        target_size = image_size)

# Converting images into numpy arraries.
img = keras.preprocessing.image.img_to_array(
        img = img,
        data_format = "channels_last",
        dtype = np.double)

filename, file_extension = os.path.splitext(p)

label = os.path.basename(filename)

#print(label)

X.append(img)
y.append(label)

print(f"Total {len(y)} images are loaded.")
#print()

X = np.array(X)
y = np.array(y)
#X = X.astype('float32')

return X, y

```

```
[32]: X, y = load_texture_dataset_xy(image_size = (128, 128),
                                    dataset_name = "Multiband_Brodatz_Texture")

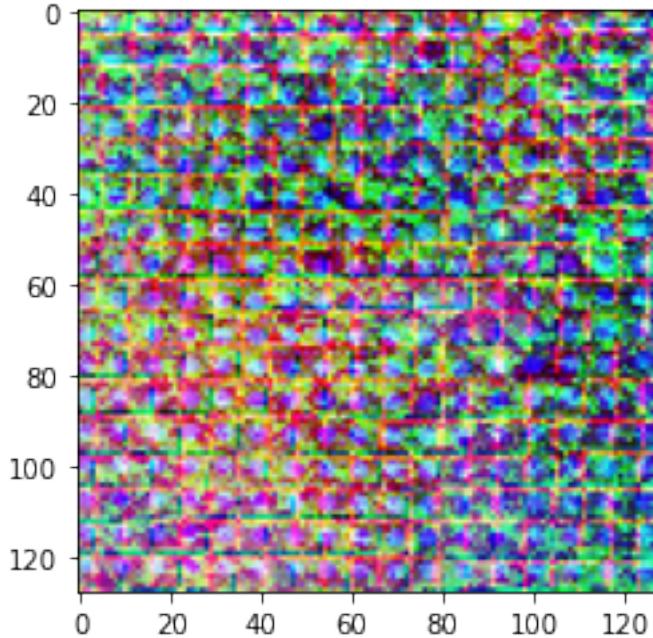
print()
print("num_of_samples: ", len(y))
print("shape: ", X.shape)

plt.imshow(np.uint8(X[0]))
plt.show()

del X, y
```

Loading dataset from path: datasets\_orig/Multiband\_Brodatz\_Texture ...  
image size: (128, 128)  
Total 154 images are loaded.

num\_of\_samples: 154  
shape: (154, 128, 128, 3)



```
[25]: def load_texture_dataset_as_dataframe(*, image_size = TEXTURE_DS_IMAGE_SIZE,
                                         dataset_name):

    X, y = load_texture_dataset_xy(image_size = image_size, dataset_name = dataset_name)

    data = {"X": list(X),
            "y": list(y)}

    df = pd.DataFrame(data)

    return df
```

```
[35]: df = load_texture_dataset_as_dataframe(image_size = (128, 128),
                                             dataset_name = "Multiband_Brodatz_Texture")

print()
print("num_of_samples: ", len(df))

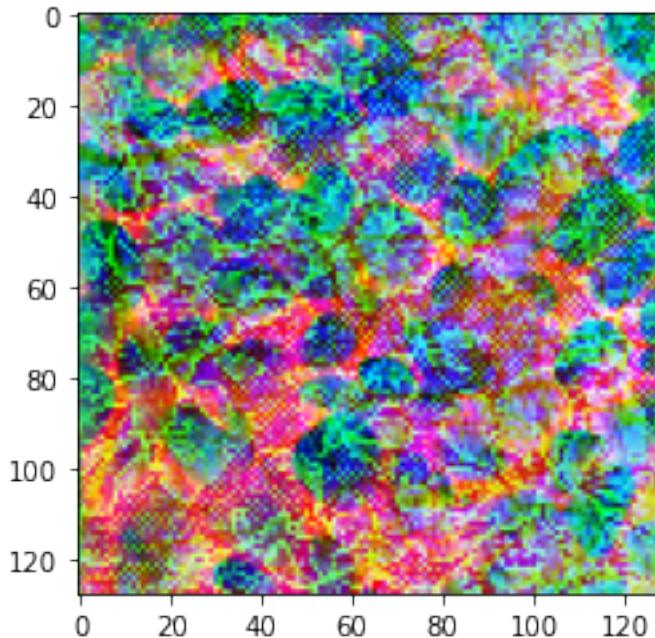
plt.imshow(np.uint8(df["X"] [2]))
plt.show()

del df
```

Loading dataset from path: datasets\_orig/Multiband\_Brodatz\_Texture ...

```
image size: (128, 128)
Total 154 images are loaded.
```

```
num_of_samples: 154
```



```
[31]: def save_texture_dataset_as_numpy(*, image_size = TEXTURE_DS_IMAGE_SIZE,
                                     dataset_name, save_prefix):

    X, y = load_texture_dataset_xy(image_size = image_size, dataset_name =
                                     dataset_name)

    np.save(f"[{save_prefix}]_X.npy", X)
    np.save(f"[{save_prefix}]_y.npy", y)
```

## 7.1 Splitting the original dataset for training and testing regarding pixels

```
[6]: import numpy as np
import pandas as pd
import os
import sys

#libpath = os.path.dirname(os.path.realpath(__file__)) + os.path.sep + ".." +
#          os.path.sep + "lib"
#libpath = os.path.dirname(os.path.realpath(__file__)) + os.path.sep + "."
#sys.path.append(libpath)
```

```

DS_DEFAULT_PIXEL_SPLIT_RATIO = 0.5

def split_dataset_by_pixel(X, y, split = DS_DEFAULT_PIXEL_SPLIT_RATIO):
    X_train = []
    y_train = []

    X_test = []
    y_test = []

    for Xi, yi in zip(X, y):
        Xi = np.array(Xi)
        #print(Xi.shape)
        #print(yi)

        height = Xi.shape[0]

        position = int(split * height)

        Xi_train = Xi[0:position, :, :]
        Xi_test = Xi[position:height, :, :]

        #print(Xi_train.shape)
        #print(Xi_test.shape)

        yi_train = yi
        yi_test = yi

        X_train.append(Xi_train)
        y_train.append(yi_train)

        X_test.append(Xi_test)
        y_test.append(yi_test)

    X_train = np.array(X_train)
    y_train = np.array(y_train)

    X_test = np.array(X_test)
    y_test = np.array(y_test)

    print()
    print("Splitting dataset into training and testing datasets ...")
    print(f"size of training dataset : {len(y_train)}")
    print(f"size of testing dataset : {len(y_test)}")
    print(f"training dataset shape: {X_train.shape}")
    print(f"testing dataset shape: {X_test.shape}")

```

```
#print()

return X_train, y_train, X_test, y_test

[43]: X, y = load_texture_dataset_xy(image_size = (128, 128),
                                    dataset_name = "Multiband_Brodatz_Texture")

X_train, y_train, X_test, y_test = split_dataset_by_pixel(X, y)

ax = plt.subplot(2, 1, 1)
plt.imshow(np.uint8(X_train[0]))

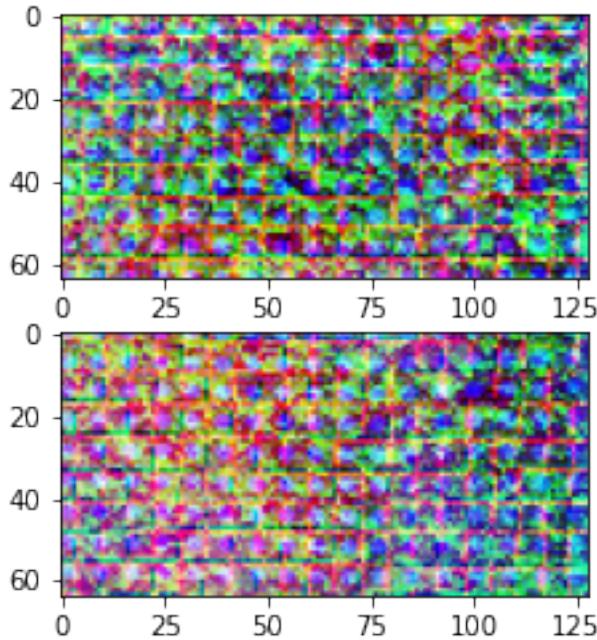
ax = plt.subplot(2, 1, 2)
plt.imshow(np.uint8(X_test[0]))

plt.show()

del X, y, X_train, y_train, X_test, y_test
```

Loading dataset from path: datasets\_orig/Multiband\_Brodatz\_Texture ...  
image size: (128, 128)  
Total 154 images are loaded.

Splitting dataset into training and testing datasets ...  
size of training dataset : 154  
size of testing dataset : 154  
training dataset shape: (154, 64, 128, 3)  
testing dataset shape: (154, 64, 128, 3)



```
[44]: def split_dataset_by_pixel_as_dataframe(X, y, split = 
    DS_DEFAULT_PIXEL_SPLIT_RATIO):

    X_train, y_train, X_test, y_test = split_dataset_by_pixel(X, y, split)

    data_train = {"X": list(X_train),
                  "y": list(y_train)}

    data_test = {"X": list(X_test),
                  "y": list(y_test)}

    df_train = pd.DataFrame(data_train)

    df_test = pd.DataFrame(data_test)

    #print("X_train.shape = ", np.array(X_train).shape)

    return df_train, df_test
```

```
[45]: X, y = load_texture_dataset_xy(image_size = (128, 128),
                                    dataset_name = "Multiband_Brodatz_Texture")

df_train, df_test = split_dataset_by_pixel_as_dataframe(X, y)

del X, y, df_train, df_test
```

```
Loading dataset from path: datasets_orig/Multiband_Brodatz_Texture ...
image size: (128, 128)
Total 154 images are loaded.
```

```
Splitting dataset into training and testing datasets ...
size of training dataset : 154
size of testing dataset : 154
training dataset shape: (154, 64, 128, 3)
testing dataset shape: (154, 64, 128, 3)
```

## 7.2 Bootstrapping new dataset for training and testing

```
[46]: import numpy as np
import pandas as pd
import os
import sys
import imageio

#libpath = os.path.dirname(os.path.realpath(__file__)) + os.path.sep + ".." + os.path.sep + "lib"
#sys.path.append(libpath)

def create_dataset_from_xy(*,
                          X,
                          y,
                          dataset_path,
                          subdir,
                          imgaug,
                          n_images_per_class):

    print()
    print("dataset_path = ", dataset_path)
    print("subdir = ", subdir)
    print("n_images_per_class = ", n_images_per_class)

    for yi in y:
        #print(yi)
        yi_path = dataset_path + os.path.sep + subdir + os.path.sep + yi

        if not os.path.exists(yi_path):
            os.makedirs(yi_path)

    for i, (Xi, yi) in enumerate(zip(X, y)):
        yi_path = dataset_path + os.path.sep + subdir + os.path.sep + yi
```

```

    print(yi, end = ' ')
    Xi = np.expand_dims(Xi, axis = 0)
    for j in range(n_images_per_class):
        Xij = imgaug(Xi)
        Xij = Xij[0]
        Xij = np.array(Xij, dtype = np.uint8)
        Xij_name = yi_path + os.path.sep + f"[yi]_{j}.png"
        #print(Xij_name)
        imageio.imwrite(Xij_name, Xij)

print()

```

```

[53]: def create_dataset( *,
                      dataset_name = "Multiband_Brodatz_Texture",
                      dataset_root = "datasets_new",

                      output_size = (128, 128),

                      n_train_images_per_class = 200,
                      n_test_images_per_class = 50,
                      seed = 1921
                      ):

dataset_path = dataset_root + os.path.sep + dataset_name
#output_shape = (*output_size, n_channels)

print("dataset_path = ", dataset_path)
print("output_size = ", output_size)
print("n_train_images_per_class = ", n_train_images_per_class)
print("n_test_images_per_class = ", n_test_images_per_class)

# loading size
image_size = (512, 512)
#n_image_channels = 3

X, y = load_texture_dataset_xy(
                    image_size = image_size,
                    dataset_name = dataset_name
                    )

X_train, y_train, X_test, y_test = split_dataset_by_pixel(X, y)

imgaug = CustomisedBootstrapImageAugmenter(
                    target_size = output_size,
                    seed = seed
                    )

```

```

if not os.path.exists(dataset_path):
    os.makedirs(dataset_path)

if not os.path.exists(dataset_path + os.path.sep + "train"):
    os.makedirs(dataset_path + os.path.sep + "train")

if not os.path.exists(dataset_path + os.path.sep + "test"):
    os.makedirs(dataset_path + os.path.sep + "test")

create_dataset_from_xy( X = X_train,
                      y = y_train,
                      dataset_path = dataset_path,
                      subdir = "train",
                      imgaug = imgaug,
                      n_images_per_class = n_train_images_per_class)

create_dataset_from_xy( X = X_test,
                      y = y_test,
                      dataset_path = dataset_path,
                      subdir = "test",
                      imgaug = imgaug,
                      n_images_per_class = n_test_images_per_class)

print()
print("Completed!")
print()

```

```
[55]: create_dataset(
            dataset_name = "Multiband_Brodatz_Texture",
            dataset_root = "datasets_new",
            output_size = (128, 128),
            n_train_images_per_class = 200,
            n_test_images_per_class = 50,
        )
```

```
dataset_path = datasets_new_tmp/Multiband_Brodatz_Texture
output_size = (128, 128)
n_train_images_per_class = 200
n_test_images_per_class = 50
```

```
Loading dataset from path: datasets_orig/Multiband_Brodatz_Texture ...
image size: (512, 512)
Total 154 images are loaded.
```

```
Splitting dataset into training and testing datasets ...
size of training dataset : 154
size of testing dataset : 154
```

```

training dataset shape: (154, 256, 512, 3)
testing dataset shape: (154, 256, 512, 3)

dataset_path = datasets_new_tmp/Multiband_Brodatz_Texture
subdir = train
n_images_per_class = 200
Dz1 Dz10 Dz100 Dz101 Dz102 Dz103 Dz104 Dz105 Dz106 Dz107 Dz108 Dz109 Dz11 Dz110
Dz111 Dz112 Dz113 Dz114 Dz115 Dz116 Dz117 Dz118 Dz119 Dz12 Dz120 Dz121 Dz122
Dz123 Dz124 Dz125 Dz126 Dz127 Dz128 Dz129 Dz13 Dz130 Dz131 Dz132 Dz133 Dz134
Dz135 Dz136 Dz137 Dz138 Dz139 Dz14 Dz140 Dz141 Dz142 Dz143 Dz144 Dz145 Dz146
Dz147 Dz148 Dz149 Dz15 Dz150 Dz151 Dz152 Dz153 Dz154 Dz16 Dz17 Dz18 Dz19 Dz2
Dz20 Dz21 Dz22 Dz23 Dz24 Dz25 Dz26 Dz27 Dz28 Dz29 Dz3 Dz30 Dz31 Dz32 Dz33 Dz34
Dz35 Dz36 Dz37 Dz38 Dz39 Dz4 Dz40 Dz41 Dz42 Dz43 Dz44 Dz45 Dz46 Dz47 Dz48 Dz49
Dz5 Dz50 Dz51 Dz52 Dz53 Dz54 Dz55 Dz56 Dz57 Dz58 Dz59 Dz6 Dz60 Dz61 Dz62 Dz63
Dz64 Dz65 Dz66 Dz67 Dz68 Dz69 Dz7 Dz70 Dz71 Dz72 Dz73 Dz74 Dz75 Dz76 Dz77 Dz78
Dz79 Dz8 Dz80 Dz81 Dz82 Dz83 Dz84 Dz85 Dz86 Dz87 Dz88 Dz89 Dz9 Dz90 Dz91 Dz92
Dz93 Dz94 Dz95 Dz96 Dz97 Dz98 Dz99

dataset_path = datasets_new_tmp/Multiband_Brodatz_Texture
subdir = test
n_images_per_class = 50
Dz1 Dz10 Dz100 Dz101 Dz102 Dz103 Dz104 Dz105 Dz106 Dz107 Dz108 Dz109 Dz11 Dz110
Dz111 Dz112 Dz113 Dz114 Dz115 Dz116 Dz117 Dz118 Dz119 Dz12 Dz120 Dz121 Dz122
Dz123 Dz124 Dz125 Dz126 Dz127 Dz128 Dz129 Dz13 Dz130 Dz131 Dz132 Dz133 Dz134
Dz135 Dz136 Dz137 Dz138 Dz139 Dz14 Dz140 Dz141 Dz142 Dz143 Dz144 Dz145 Dz146
Dz147 Dz148 Dz149 Dz15 Dz150 Dz151 Dz152 Dz153 Dz154 Dz16 Dz17 Dz18 Dz19 Dz2
Dz20 Dz21 Dz22 Dz23 Dz24 Dz25 Dz26 Dz27 Dz28 Dz29 Dz3 Dz30 Dz31 Dz32 Dz33 Dz34
Dz35 Dz36 Dz37 Dz38 Dz39 Dz4 Dz40 Dz41 Dz42 Dz43 Dz44 Dz45 Dz46 Dz47 Dz48 Dz49
Dz5 Dz50 Dz51 Dz52 Dz53 Dz54 Dz55 Dz56 Dz57 Dz58 Dz59 Dz6 Dz60 Dz61 Dz62 Dz63
Dz64 Dz65 Dz66 Dz67 Dz68 Dz69 Dz7 Dz70 Dz71 Dz72 Dz73 Dz74 Dz75 Dz76 Dz77 Dz78
Dz79 Dz8 Dz80 Dz81 Dz82 Dz83 Dz84 Dz85 Dz86 Dz87 Dz88 Dz89 Dz9 Dz90 Dz91 Dz92
Dz93 Dz94 Dz95 Dz96 Dz97 Dz98 Dz99

```

Completed!

## 8 Optimising memory use

[346]: `import gc`

```
gc.collect()
```

[346]: 64546

### 8.1 Visualising the bootstrapped datasets

### 8.1.1 Plotting dataset samples

```
[74]: import numpy as np
import matplotlib.pyplot as plt

""" We plot the sample images from the dataset.
"""

def plot_dataset_samples(*,
                        df,
                        nrows = 4,
                        ncols = 4,
                        subfig_height = 4,
                        subfig_width = 4,
                        ):
    """ Plotting the samples

    """

    figsize = (ncols * subfig_width, nrows * subfig_height)
    #figsize = (15, 15)

    """ Sampling

    """
    dfs = df.sample(n = nrows * ncols, replace = False)

    Xs = np.array(dfs["X"], dtype = object)
    ys = np.array(dfs["y"], dtype = object)

    fig, axes = plt.subplots(nrows = nrows,
                            ncols = ncols,
                            figsize = figsize)

    """ plt.subplots_adjust(left = 0.02,
                         right = 0.97,
                         bottom = 0.05,
                         hspace = 0.1,
                         wspace = 0.1,
                         top = 0.95)
    """


```

```

axes_flat = axes.flat

for row in range(nrows):
    for col in range(ncols):

        imgid = row * ncols + col

        img = Xs[imgid]
        #img = np.array(img, dtype = np.uint8) #print("img.shape", img.
        ↪shape)

        label = ys[imgid]

        ax = axes_flat[row * ncols + col]

        ax.set_xticks([])
        ax.set_yticks([])

        ax.imshow(img, vmin = 0, vmax = 1)
        #ax.imshow(img)
        ax.title.set_text(f"{label} #{imgid}")

plt.show()
plt.close()

```

```

[75]: df = load_texture_dataset_as_dataframe(dataset_name =
    ↪"Multiband_Brodatz_Texture")

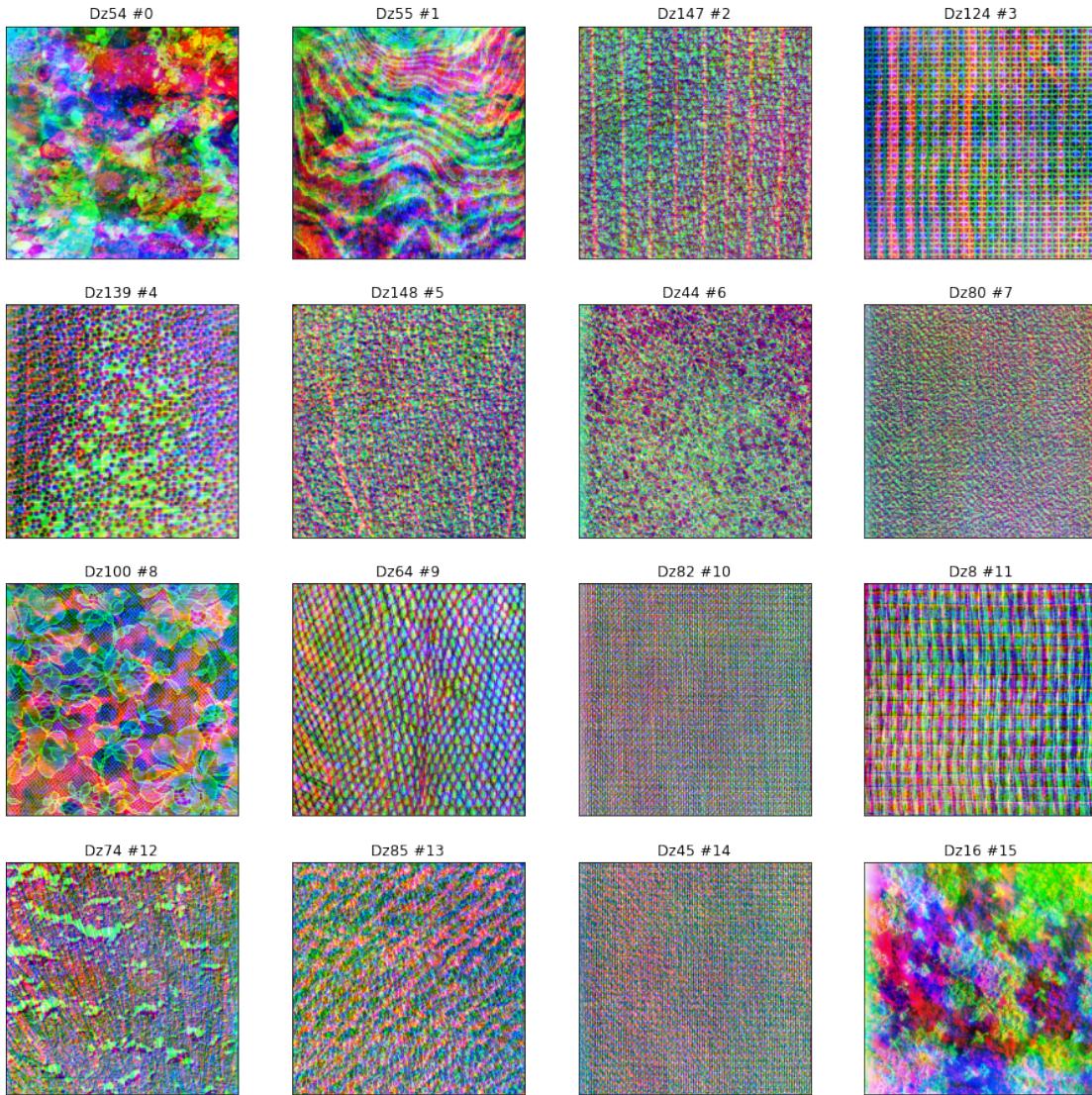
df["X"] /= 255

plot_dataset_samples(    df = df,
                        nrows = 4,
                        ncols = 4,
                        subfig_height = 4,
                        subfig_width = 4,
                        )

del df

```

Loading dataset from path: datasets\_orig/Multiband\_Brodatz\_Texture ...  
image size: (512, 512)  
Total 154 images are loaded.



### 8.1.2 Plotting augmented various samples

```
[355]: import numpy as np
import matplotlib.pyplot as plt
import os

def plot_augmented_samples(*,
                          X,
                          y,
                          img_augmenter,
                          nrows = 2,
                          ncols = 4,
                          subfig_width = 4,
```

```

        subfig_height = 4,
        savename = None
    ):

#X, y = next(iter(dataset))

#X,y = load_mbt_dataset()

#X_train, y_train, X_test, y_test = split_mbt_dataset(X, y)

#img_augmenter = CustomisedUtilImageAugmenter()

figsize = (ncols * subfig_width, nrows * subfig_height)

fig, axes = plt.subplots(nrows = nrows,
                        ncols = ncols,
                        figsize = figsize)

plt.subplots_adjust(left = 0.02,
                    right = 0.98,
                    bottom = 0.02,
                    top = 0.98)

""" plt.subplots_adjust(left = 0.02,
                     right = 0.97,
                     bottom = 0.05,
                     hspace = 0.1,
                     wspace = 0.1,
                     top = 0.95)
"""

axes_flat = axes.flat

for row in range(nrows):
    for col in range(ncols):

        imgid = row * ncols + col

        Xi = X[imgid, :, :, :]
        #Xi = X_train[3, :, :, :]
        Xi = Xi.copy()
        Xi = np.expand_dims(Xi, axis = 0)
        Xi = img_augmenter(Xi)
        #Xi = np.array(Xi, dtype = np.uint8)

```

```

    img = Xi[0, :, :, :]

    #img = X_train[imgid, :, :, :]
    #img = np.array(img, dtype = np.uint8) #print("img.shape", img.
    ↪shape)

    label = y[imgid]

    ax = axes.flat[row * ncols + col]

    ax.set_xticks([])
    ax.set_yticks([])

    ax.imshow(img, vmin = 0, vmax = 1.0)
    #ax.imshow(img)

    ax.title.set_text(f"{{label}} #{{imgid}}")

if not os.path.exists("figures"):
    os.makedirs("figures")

if savename:

    figpath = "figures" + os.path.sep + savename

    plt.savefig(figpath,
                dpi=600,
                format='pdf')

    print("saved to: ", figpath)

plt.show()
plt.close()

```

```

[356]: img_augmenter = CustomisedBootstrapImageAugmenter(
            target_size = (128, 128),
            seed = 2113
        )

X, y = load_texture_dataset_xy(dataset_name = "Multiband_Brodatz_Texture")

plot_augmented_samples(
            X = X / 255,
            y = y,

```

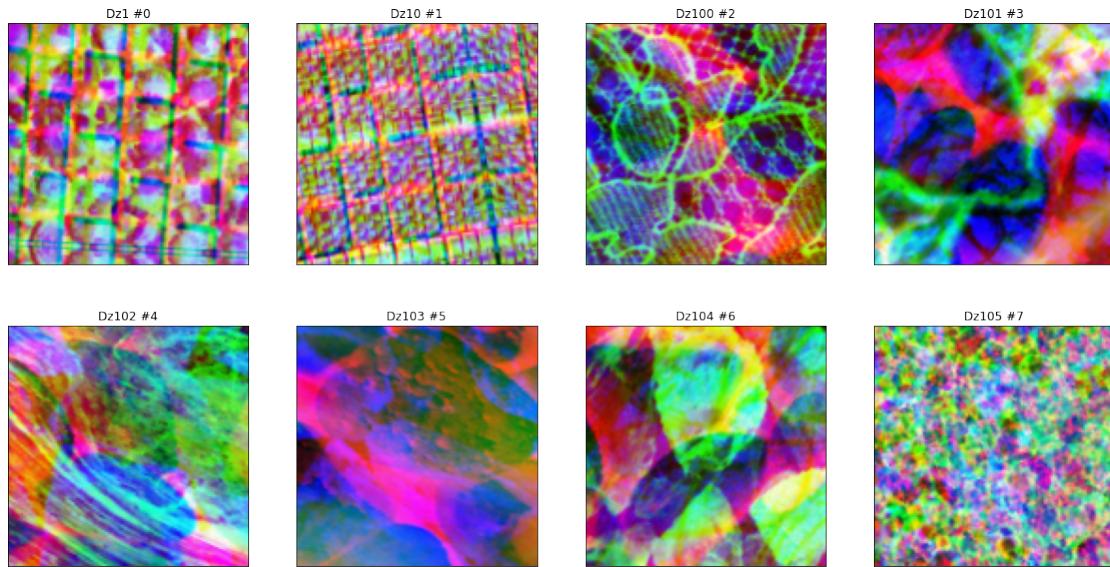
```

        img_augmenter = img_augmenter,
        nrows = 2,
        ncols = 4,
        subfig_width = 4,
        subfig_height = 4,
        savename = "fig_augmented_samples.pdf"
    )

del img_augmenter, X, y

```

Loading dataset from path: datasets\_orig/Multiband\_Brodatz\_Texture ...  
image size: (512, 512)  
Total 154 images are loaded.  
saved to: figures/fig\_augmented\_samples.pdf



```
[357]: def plot_augmented_samples_for_one_image(*,
                                             X,
                                             y,
                                             img_augmenter,
                                             nrows = 2,
                                             ncols = 4,
                                             subfig_width = 4,
                                             subfig_height = 4,
                                             savename = None,
                                             ):

```

```

#X, y = next(iter(dataset))

#X,y = load_mbt_dataset()
#X_train, y_train, X_test, y_test = split_mbt_dataset(X, y)



```

```



```

```

[368]: img_augmenter = CustomisedBootstrapImageAugmenter(
            target_size = (128, 128),
            seed = 2113
        )

X, y = load_texture_dataset_xy(dataset_name = "Multiband_Brodatz_Texture")

plot_augmented_samples_for_one_image(
            X = X / 255,
            y = y,
            img_augmenter = img_augmenter,
            nrows = 2,
            ncols = 4,

```

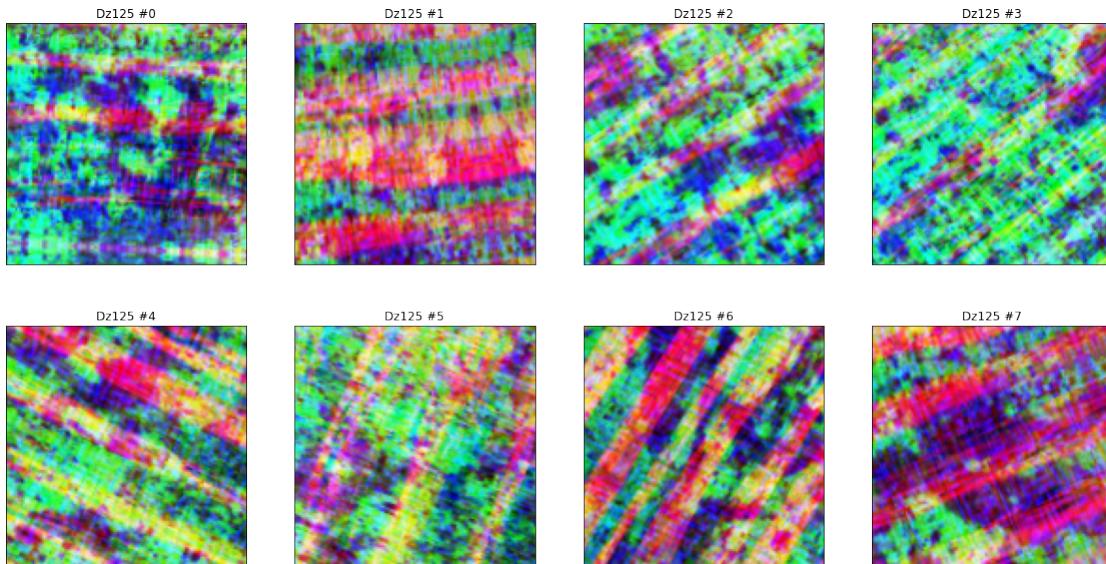
```

        subfig_width = 4,
        subfig_height = 4,
        savename = "fig_augmented_samples_for_one_image.
→pdf"
    )

del img_augmenter, X, y

```

Loading dataset from path: datasets\_orig/Multiband\_Brodatz\_Texture ...  
image size: (512, 512)  
Total 154 images are loaded.  
saved to: figures/fig\_augmented\_samples\_for\_one\_image.pdf



## 9 Loading the final training and testing datasets

### 9.1 Loading datasets

```
[4]: train_dataset_ = tf.keras.preprocessing.image_dataset_from_directory(
    "datasets_new/Multiband_Brodatz_Texture/train",
    image_size = image_size,
    batch_size = batch_size,
    interpolation = "bilinear",
    color_mode = "rgb",
    shuffle = True,
    seed = 2113
)
```

```
Found 30800 files belonging to 154 classes.
```

```
[5]: test_dataset_ = tf.keras.preprocessing.image_dataset_from_directory(  
        "datasets_new/Multiband_Brodatz_Texture/test",  
        image_size = image_size,  
        batch_size = batch_size,  
        interpolation = "bilinear",  
        color_mode = "rgb",  
        shuffle = True,  
        seed = 3371  
)
```

```
Found 7700 files belonging to 154 classes.
```

```
[6]: n_classes = len(train_dataset_.class_names)  
print("n_classes = ", n_classes)
```

```
n_classes = 154
```

```
[7]: n_train_samples = len(train_dataset_) * batch_size  
n_train_batches = len(train_dataset_)  
  
print("n_train_samples = ", n_train_samples)  
print("n_train_batches = ", n_train_batches)
```

```
n_train_samples = 30848  
n_train_batches = 241
```

```
[8]: n_test_samples = len(test_dataset_) * batch_size  
n_test_batches = len(test_dataset_)  
  
print("n_test_samples = ", n_test_samples)  
print("n_test_batches = ", n_test_batches)
```

```
n_test_samples = 7808  
n_test_batches = 61
```

```
[9]: n_samples = n_train_samples + n_test_samples  
n_batches = n_train_batches + n_test_batches  
  
print("n_samples = ", n_samples)  
print("n_batches = ", n_batches)
```

```
n_samples = 38656  
n_batches = 302
```

## 9.2 Normalising datasets and setting data prefetching

```
[10]: train_dataset = train_dataset_.map(lambda x, y: (x / 255., y))
train_dataset = train_dataset.prefetch(buffer_size = tf.data.AUTOTUNE)

test_dataset = test_dataset_.map(lambda x, y: (x / 255., y))
test_dataset = test_dataset.prefetch(buffer_size = tf.data.AUTOTUNE)
```

## 9.3 Verifying loaded dataset

```
[92]: x1 = next(iter(train_dataset))
x2 = x1[0]
x2.shape
```

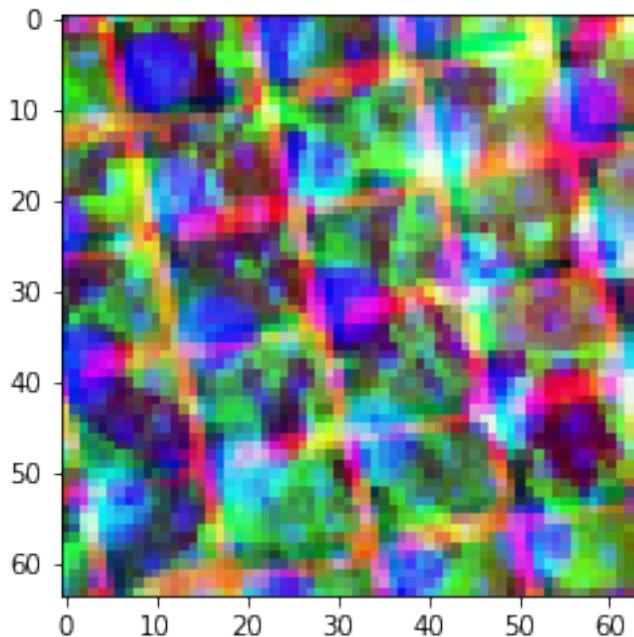
```
[92]: TensorShape([128, 64, 64, 3])
```

```
[93]: import matplotlib.pyplot as plt

img = next(iter(train_dataset))[0][0]

plt.imshow(img, vmin=0, vmax=1)
plt.show()
```



## 10 Image augmentation model

## 10.1 Random color affine transformation layer

Random color affine transformation is crucial to avoid trivial solution of merely learning classification from color histogram information!

```
[11]: # We make it working with keras model save and load
# ref:
# https://keras.io/guides/serialization_and_saving/
# https://stackoverflow.com/questions/62280161/
#       ↳saving-keras-models-with-custom-layers
# https://keras.io/examples/vision/semisupervised_simclr/

import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers.experimental import preprocessing as u
#       ↳keras_preprocessing_layers

class CustomisedRandomColorAffineLayer(keras.layers.Layer):

    def __init__(self,
                 brightness = 0,
                 jitter = 0,
                 name = 'color_affine',
                 **kwargs):
        self.brightness = brightness
        self.jitter = jitter

        super(CustomisedRandomColorAffineLayer, self).__init__(**kwargs)

    def get_config(self):

        config = super().get_config()

        config['brightness'] = self.brightness
        config['jitter'] = self.jitter

        #print("config = ", config)

        return config

    #@classmethod
    #def from_config(cls, config):
    #    return cls(**config)

    def call(self, images, training = True):
```

```

if training:
    batch_size = tf.shape(images)[0]

    # For each pixels, we randomly adjust their brightnesses.
    brightness_scales = 1 + tf.random.uniform(
        (batch_size, 1, 1, 1),
        minval = -self.brightness,
        maxval = self.brightness
    )

    # Different for all colors
    jitter_matrices = tf.random.uniform(
        (batch_size, 1, 3, 3),
        minval = -self.jitter,
        maxval = self.jitter
    )

    color_transforms = (
        tf.eye(3, batch_shape = [batch_size, 1]) * ↳
    ↳brightness_scales
        + jitter_matrices
    )

    images = tf.clip_by_value(tf.matmul(images, color_transforms), 0, 1)

return images

```

## 10.2 Image augmentation model for train

```

[12]: import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers.experimental import preprocessing as ↳
    ↳keras_preprocessing_layers

class CustomisedTrainImageAugmenter(keras.layers.Layer):
    def __init__(self,
                 min_area,
                 brightness,
                 jitter,
                 name = 'train_imgaug_layer',
                 **kwargs):

        #self.image_shape = image_shape
        self.min_area = min_area
        self.brightness = brightness

```

```

    self.jitter = jitter

#self.zoom_factor = 1.0 - tf.sqrt(min_area)
self.zoom_factor = 1.0 - np.sqrt(min_area)

self.model = keras.Sequential(
[
    #keras.layers.InputLayer(input_shape = image_shape),
    #keras_preprocessing_layers.Rescaling(1 / 255),
    keras.preprocessing_layers.RandomFlip("horizontal_and_vertical"),
    keras.preprocessing_layers.RandomTranslation(
        self.zoom_factor / 2,
        self.zoom_factor / 2
    ),
    keras.preprocessing_layers.RandomZoom(
        (-self.zoom_factor, 0.0),
        (-self.zoom_factor, 0.0)
    ),
    CustomisedRandomColorAffineLayer(
        brightness = brightness,
        jitter = jitter,
        name = "color_affine_for_train"
    ),
],
name = "train_imgaug_model"
)

super(CustomisedTrainImageAugmenter, self).__init__(name = name, **kwargs)

def get_config(self):
    config = super().get_config()

    #config['image_shape'] = self.image_shape
    config['min_area'] = self.min_area
    config['brightness'] = self.brightness
    config['jitter'] = self.jitter
    config['zoom_factor'] = self.zoom_factor
    config['model'] = self.model

    #print(config)

    return config

def call(self, images, training = True):
    if training:

```

```

        outputs = self.model(images)
    else:
        outputs = images

    return outputs

@classmethod
def from_config(cls, config):
    return cls(**config)

```

## 11 Visualising image augmentation

### 11.1 Plotting augmented various samples

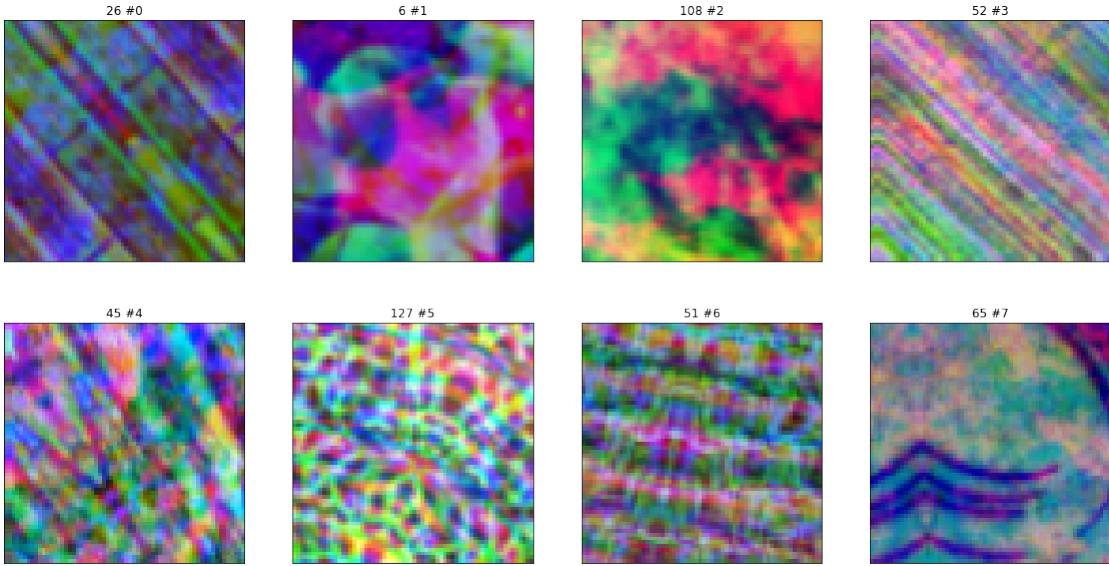
```
[371]: img_augmenter = CustomisedTrainImageAugmenter(**imgaug_params)

X, y = next(iter(train_dataset))

X = X.numpy().astype("float32")

plot_augmented_samples(
    X = X,
    y = y,
    img_augmenter = img_augmenter,
    nrows = 2,
    ncols = 4,
    subfig_width = 4,
    subfig_height = 4,
    savename = "fig_augmented_samples_for_train.pdf"
)
del img_augmenter, X, y
```

saved to: figures/fig\_augmented\_samples\_for\_train.pdf



## 11.2 Plotting augmented samples for one image

```
[372]: img_augmenter = CustomisedTrainImageAugmenter(**imgaug_params)

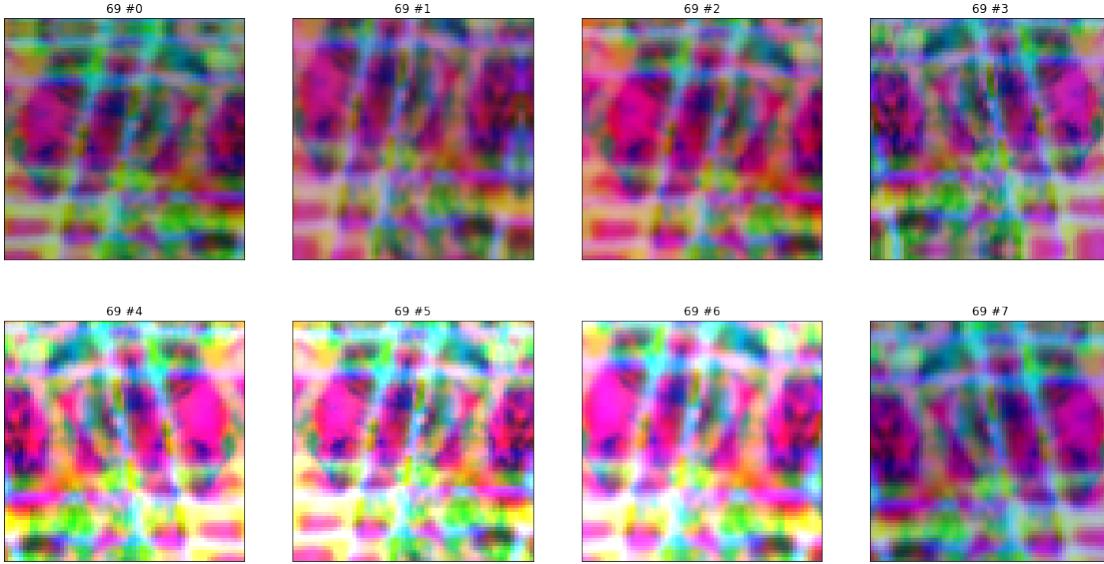
X, y = next(iter(train_dataset))

X = X.numpy().astype("float32")

plot_augmented_samples_for_one_image(
    X = X,
    y = y,
    img_augmenter = img_augmenter,
    nrows = 2,
    ncols = 4,
    subfig_width = 4,
    subfig_height = 4,
    savename =_
    →"fig_augmented_samples_for_one_image_for_train.pdf"
    )

del img_augmenter, X, y
```

saved to: figures/fig\_augmented\_samples\_for\_one\_image\_for\_train.pdf



## 12 Building supervised learning model with specified encoder

```
[277]: def build_supervised_model(*,
                               learning_rate = 0.001,
                               image_shape,
                               n_classes,
                               encoder,
                               image_augmenter,
                               name
                               ):

    newhead = keras.Sequential(
        [
            keras.layers.Dense(512, activation="relu", name = "hidden_1"),
            keras.layers.Dense(512, activation="relu", name = "hidden_2"),
            keras.layers.Dense(n_classes, name = "output"),
        ],
        name = "newhead"
    )

    model = keras.Sequential(
        [
            keras.layers.InputLayer(input_shape = image_shape),
```

```

        image_augmenter, )
    ↵#CustomisedTrainImageAugmenter(**imgaug_params),
        encoder,
        newhead
    ],
    name = name,
)

model.compile(
    optimizer = keras.optimizers.Adam(learning_rate = learning_rate),
    loss = keras.losses.SparseCategoricalCrossentropy(from_logits = ↵
True),
    metrics = [
        keras.metrics.SparseCategoricalAccuracy(name = ↵
"accuracy"),
        keras.metrics.SparseTopKCategoricalAccuracy(5, ↵
name="accuracy_top-5"),
    ],
)
)

#print()
#newhead.summary()

#print()
#encoder.summary()

#print()
#model.summary()

return model

```

## 13 A framework of training, evaluating and saving

```
[21]: import os
import pickle

def train_evaluate_and_save_models(*,
        model,
        encoder,
        train_dataset,
        test_dataset,
        epochs,
        savename,
        callbacks = []
):

```

```

checkpoint_filepath = "checkpoints" + os.path.sep + savename

if not os.path.exists(checkpoint_filepath):
    os.makedirs(checkpoint_filepath)

checkpoint_callback = keras.callbacks.ModelCheckpoint(
    checkpoint_filepath,
    monitor="val_accuracy",
    save_best_only = False,
    save_weights_only = True,
)

print()
print("Training ...")
print()

history_train = model.fit(
    train_dataset,
    epochs = epochs,
    validation_data = test_dataset,
    verbose = 1,
    callbacks = callbacks,
)

#loading the checkpoint
#model.load_weights(checkpoint_filepath)

savepath = "models" + os.path.sep + savename

if not os.path.exists(savepath):
    os.makedirs(savepath)

print()
print("Saving models ...")
print()

if encoder:
    encoder.save(savepath + os.path.sep + "encoder.saved_model")

#a tensorflow bug, must call the model first then
_ = model(next(iter(train_dataset))[0])

```

```

model.save(savepath + os.path.sep + "main.saved_model")

print()
print("Saving training history data ...")
print()

if not os.path.exists("results" + os.path.sep + savename):
    os.makedirs("results" + os.path.sep + savename)

pickle.dump(history_train.history,
            open("results" + os.path.sep + savename + os.path.sep + "history_train.pickle", "wb"))

print()
print("Results:")
print()
history = history_train.history
for k in history:

    if "loss" in k:
        continue

    max_v = float(max(history[k]))

    print(f"Maximal {k}: {max_v:.5f}")

print()
print("-- Completed --")
print()

```

## 14 The facilities of plotting results

```
[15]: import pickle
import matplotlib.pyplot as plt
from matplotlib import cm
from itertools import cycle

#from matplotlib.colors import ListedColormap, LinearSegmentedColormap

def plot_train_results(
                        resultfile,
                        title,
                        ylabel,
                        keys,
                        labels,
```

```

        savename = None,
        legend_location = "best",
        normalise = False
    ):

if not os.path.exists(resultfile):
    return

#fig_width = 10.
#fig_height = fig_width*0.5
#
#fig = plt.figure(figsize=(fig_width, fig_height))
#
#
#plt.subplots_adjust(left = 0.1,
#                     right = 0.9,
#                     bottom = 0.15,
#                     top = 0.9)

fig_width = 8.5
fig_height = fig_width*0.5

fig = plt.figure(figsize=(fig_width, fig_height))

plt.subplots_adjust(left = 0.1,
                    right = 0.95,
                    bottom = 0.15,
                    top = 0.95)

history = pickle.load(open(resultfile, "rb"))

n = len(keys)
#colours = iter(cm.rainbow(np.linspace(0, 1, n)))
colours = iter(cm.Set1(np.linspace(0, 1, n)))

markers = cycle(["v", "^", "s", "<", ">", "D"])

for k, label in zip(keys, labels):
    if k not in history:
        print(f"key {k} not found")
        continue

    v = history[k]

```

```

if normalise:
    v = np.array(v) / max(v)

c = next(colours)
m = next(markers)

# plt.plot(v, color = c, label = label, linewidth = 2)
plt.plot(
    v,
    color = c,
    label = label,
    linewidth = 2.5,
    # linestyle = ls,
    marker = m,
    markersize = 9
)

plt.xlabel("# of epochs", fontsize = 12)
plt.ylabel(ylabel, fontsize = 12)

k = list(keys)[0]
v = history[k]
plt.xlim(0, len(v) - 1)
plt.ylim(0, 1.3)

xticks = np.linspace(0,
                     len(v) - 1,
                     num = len(v),
                     endpoint = True,
                     dtype = int)
xlabels = xticks

plt.xticks(xticks, xlabels)
plt.yticks([0, 0.25, 0.5, 0.75, 1], [0, 0.25, 0.5, 0.75, 1])

plt.grid()

plt.legend(
    loc = legend_location, #'upper center',
    fontsize = 12,
    ncol = 3,
    frameon = False)

if not os.path.exists("figures"):

```

```

os.makedirs("figures")

if savename:
    figpath = "figures" + os.path.sep + savename
    plt.savefig(figpath,
                dpi=600,
                format='pdf')

    print("saved to: ", figpath)

plt.title(title)

plt.show()

```

## 15 Baseline: Supervised learning with simple encoder

### 15.1 Building simple encoder

```
[16]: def build_simple_encoder(image_shape):
    return keras.Sequential(
        [
            keras.layers.InputLayer(input_shape=image_shape),

            keras.layers.Conv2D(32, kernel_size=3, strides=2, activation="relu"),
            keras.layers.Conv2D(64, kernel_size=3, strides=2, activation="relu"),
            keras.layers.BatchNormalization(),
            keras.layers.Dropout(0.3),

            keras.layers.Conv2D(128, kernel_size=3, strides=2, activation="relu"),
            keras.layers.Conv2D(256, kernel_size=3, strides=2, activation="relu"),
            keras.layers.BatchNormalization(),
            keras.layers.Dropout(0.3),

            keras.layers.Flatten(),
        ],
        name = "encoder_simple",
    )
```

## 15.2 Building model

```
[279]: baseline_simple_imgaug = CustomisedTrainImageAugmenter(**imgaug_params)
baseline_simple_encoder = build_simple_encoder(image_shape)

baseline_simple_model = build_supervised_model(
    image_shape = image_shape,
    n_classes = n_classes,
    encoder = baseline_encoder_simple,
    image_augmenter = baseline_simple_imgaug,
    name = "baseline_simple_model",
    learning_rate = 0.001)
```

## 15.3 Training from scratch

```
[280]: train_evaluate_and_save_models(
    model = baseline_simple_model,
    encoder = baseline_simple_encoder,
    train_dataset = train_dataset,
    test_dataset = test_dataset,
    epochs = n_epochs,
    savename = "baseline_simple",
)
```

Training ...

```
Epoch 1/20
241/241 [=====] - 53s 214ms/step - loss: 0.7938 -
accuracy: 0.8040 - accuracy_top-5: 0.9220 - val_loss: 1.1209 - val_accuracy:
0.7174 - val_accuracy_top-5: 0.9369
Epoch 2/20
241/241 [=====] - 52s 214ms/step - loss: 0.1690 -
accuracy: 0.9431 - accuracy_top-5: 0.9982 - val_loss: 0.8526 - val_accuracy:
0.8052 - val_accuracy_top-5: 0.9577
Epoch 3/20
241/241 [=====] - 58s 238ms/step - loss: 0.1324 -
accuracy: 0.9569 - accuracy_top-5: 0.9992 - val_loss: 1.2039 - val_accuracy:
0.7466 - val_accuracy_top-5: 0.9403
Epoch 4/20
241/241 [=====] - 60s 247ms/step - loss: 0.1204 -
accuracy: 0.9615 - accuracy_top-5: 0.9996 - val_loss: 0.9634 - val_accuracy:
0.7766 - val_accuracy_top-5: 0.9629
Epoch 5/20
241/241 [=====] - 51s 211ms/step - loss: 0.1088 -
accuracy: 0.9644 - accuracy_top-5: 0.9994 - val_loss: 2.3881 - val_accuracy:
0.6212 - val_accuracy_top-5: 0.8921
Epoch 6/20
```

```
241/241 [=====] - 51s 211ms/step - loss: 0.1056 -  
accuracy: 0.9658 - accuracy_top-5: 0.9998 - val_loss: 1.2140 - val_accuracy:  
0.7464 - val_accuracy_top-5: 0.9505  
Epoch 7/20  
241/241 [=====] - 51s 209ms/step - loss: 0.0921 -  
accuracy: 0.9697 - accuracy_top-5: 0.9996 - val_loss: 0.9403 - val_accuracy:  
0.8036 - val_accuracy_top-5: 0.9670  
Epoch 8/20  
241/241 [=====] - 51s 210ms/step - loss: 0.0813 -  
accuracy: 0.9739 - accuracy_top-5: 0.9998 - val_loss: 1.3903 - val_accuracy:  
0.7338 - val_accuracy_top-5: 0.9395  
Epoch 9/20  
241/241 [=====] - 51s 210ms/step - loss: 0.0798 -  
accuracy: 0.9730 - accuracy_top-5: 0.9997 - val_loss: 1.0285 - val_accuracy:  
0.7888 - val_accuracy_top-5: 0.9647  
Epoch 10/20  
241/241 [=====] - 51s 211ms/step - loss: 0.0777 -  
accuracy: 0.9748 - accuracy_top-5: 0.9997 - val_loss: 0.7458 - val_accuracy:  
0.8543 - val_accuracy_top-5: 0.9660  
Epoch 11/20  
241/241 [=====] - 51s 211ms/step - loss: 0.0753 -  
accuracy: 0.9761 - accuracy_top-5: 0.9999 - val_loss: 0.9297 - val_accuracy:  
0.8164 - val_accuracy_top-5: 0.9635  
Epoch 12/20  
241/241 [=====] - 51s 211ms/step - loss: 0.0823 -  
accuracy: 0.9734 - accuracy_top-5: 0.9998 - val_loss: 1.3488 - val_accuracy:  
0.7304 - val_accuracy_top-5: 0.9534  
Epoch 13/20  
241/241 [=====] - 51s 212ms/step - loss: 0.0672 -  
accuracy: 0.9773 - accuracy_top-5: 0.9999 - val_loss: 1.7099 - val_accuracy:  
0.7140 - val_accuracy_top-5: 0.9240  
Epoch 14/20  
241/241 [=====] - 51s 211ms/step - loss: 0.0652 -  
accuracy: 0.9793 - accuracy_top-5: 0.9998 - val_loss: 0.8247 - val_accuracy:  
0.8240 - val_accuracy_top-5: 0.9701  
Epoch 15/20  
241/241 [=====] - 52s 213ms/step - loss: 0.0661 -  
accuracy: 0.9791 - accuracy_top-5: 0.9998 - val_loss: 1.0479 - val_accuracy:  
0.7913 - val_accuracy_top-5: 0.9618  
Epoch 16/20  
241/241 [=====] - 52s 213ms/step - loss: 0.0640 -  
accuracy: 0.9786 - accuracy_top-5: 0.9998 - val_loss: 0.8423 - val_accuracy:  
0.8369 - val_accuracy_top-5: 0.9668  
Epoch 17/20  
241/241 [=====] - 51s 210ms/step - loss: 0.0557 -  
accuracy: 0.9817 - accuracy_top-5: 0.9998 - val_loss: 1.1251 - val_accuracy:  
0.7791 - val_accuracy_top-5: 0.9640  
Epoch 18/20
```

```
241/241 [=====] - 51s 211ms/step - loss: 0.0579 -
accuracy: 0.9815 - accuracy_top-5: 0.9998 - val_loss: 0.7612 - val_accuracy:
0.8425 - val_accuracy_top-5: 0.9742
Epoch 19/20
241/241 [=====] - 51s 211ms/step - loss: 0.0656 -
accuracy: 0.9794 - accuracy_top-5: 0.9998 - val_loss: 0.9839 - val_accuracy:
0.8022 - val_accuracy_top-5: 0.9623
Epoch 20/20
241/241 [=====] - 53s 217ms/step - loss: 0.0590 -
accuracy: 0.9810 - accuracy_top-5: 0.9999 - val_loss: 0.9123 - val_accuracy:
0.8231 - val_accuracy_top-5: 0.9705
```

Saving models ...

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.

```
INFO:tensorflow:Assets written to:
models/baseline_simple/encoder.saved_model/assets
INFO:tensorflow:Assets written to:
models/baseline_simple/encoder.saved_model/assets
INFO:tensorflow:Assets written to:
models/baseline_simple/main.saved_model/assets
INFO:tensorflow:Assets written to:
models/baseline_simple/main.saved_model/assets
```

Saving training history data ...

Results:

```
Maximal accuracy: 0.98172
Maximal accuracy_top-5: 0.99994
Maximal val_accuracy: 0.85429
Maximal val_accuracy_top-5: 0.97416
```

-- Completed --

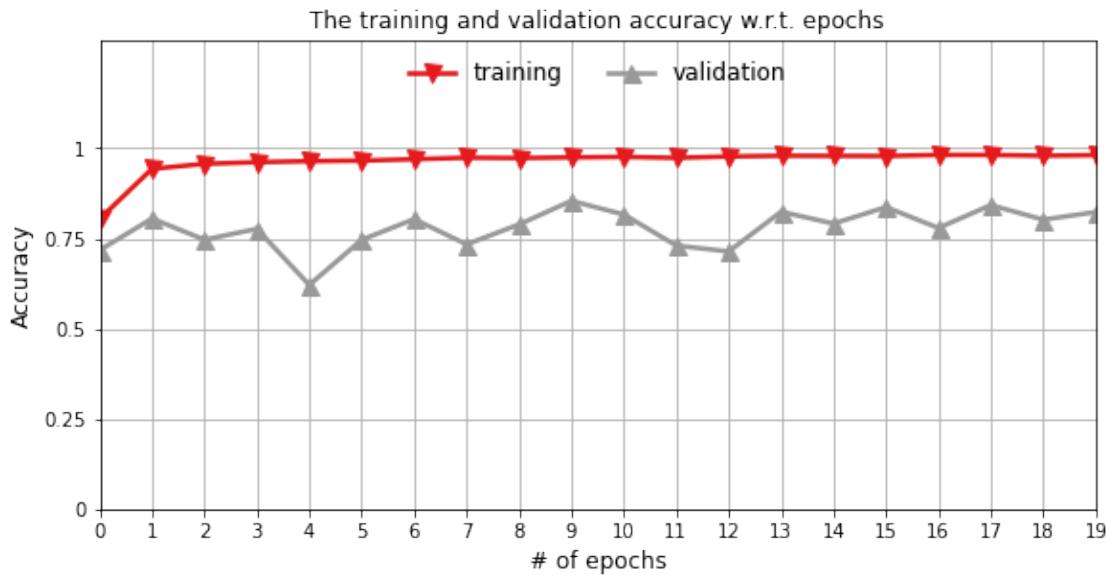
## 15.4 Testing of loading models

```
[ ]: test_encoder = keras.models.load_model("models/baseline_simple/encoder.  
→saved_model")  
test_model = keras.models.load_model("models/baseline_simple/main.saved_model")  
  
del test_encoder  
del test_model  
  
[176]: history = pickle.load(open(b"results/baseline_simple/history_train.pickle",  
→"rb"))  
del history
```

## 15.5 Plotting results

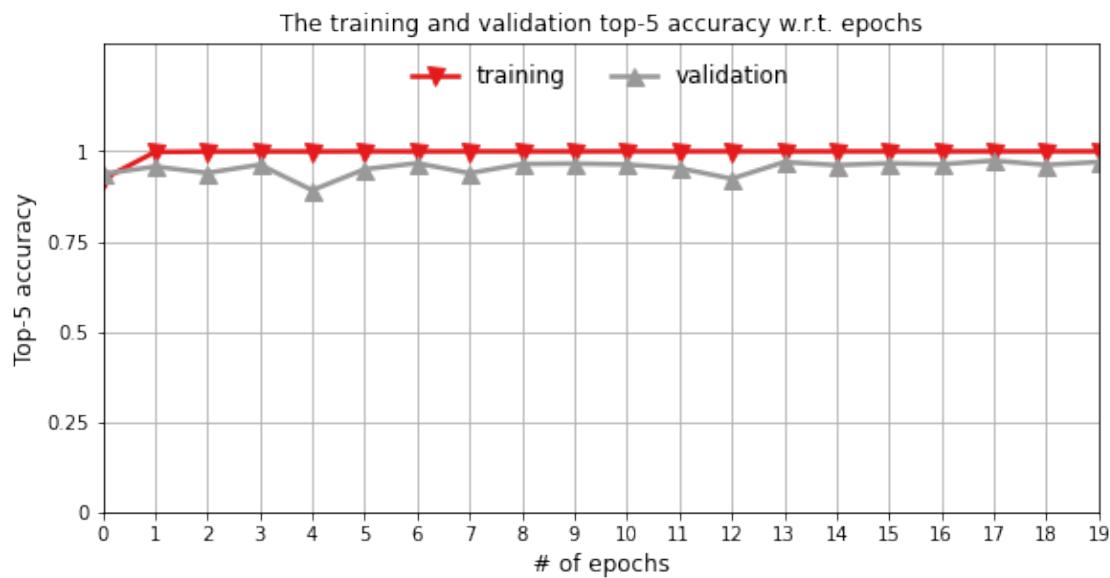
```
[16]: plot_train_results(  
        resultfile = "results/baseline_simple/history_train.pickle",  
        title = "The training and validation accuracy w.r.t. epochs",  
        ylabel = "Accuracy",  
        keys = ["accuracy", "val_accuracy"],  
        labels = ["training", "validation"],  
        savename = "fig_baseline_simple_accuracy.pdf",  
        legend_location = "upper center",  
        normalise = False  
)
```

saved to: figures/fig\_baseline\_simple\_accuracy.pdf



```
[17]: plot_train_results(
        resultfile = "results/baseline_simple/history_train.pickle",
        title = "The training and validation top-5 accuracy w.r.t. epochs",
        ylabel = "Top-5 accuracy",
        keys = ["accuracy_top-5", "val_accuracy_top-5"],
        labels = ["training", "validation"],
        savename = "fig_baseline_simple_accuracy_top5.pdf",
        legend_location = "upper center",
        normalise = False
    )
```

saved to: figures/fig\_baseline\_simple\_accuracy\_top5.pdf



## 16 Baseline: Supervised learning with VGG16

### 16.1 Building VGG16 encoder

```
[281]: from tensorflow.keras.applications import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input as vgg16_preprocess_input

def build_vgg16_encoder(image_shape):
    inputs = keras.Input(shape = image_shape, name = "inputs")
```

```

#x = keras.layers.Lambda(vgg16_preprocess_input)(inputs)

model_orig = VGG16(
    include_top = False,      # no pre-defined head
    input_tensor = inputs,    # input tensor
    input_shape = image_shape, # input shape
    weights = "imagenet",
    pooling = "max"          #max avg
)

outputs = model_orig.layers[-1].output
#outputs = keras.layers.GlobalAveragePooling2D()(outputs)

model_new = keras.Model(inputs = inputs, outputs = outputs, name = "vgg16_encoder")

return model_new

```

## 16.2 Building model

```
[282]: baseline_vgg16_imgaug = CustomisedTrainImageAugmenter(**imgaug_params)
baseline_vgg16_encoder = build_vgg16_encoder(image_shape)
baseline_vgg16_encoder.trainable = False

baseline_vgg16_model = build_supervised_model(
    image_shape = image_shape,
    n_classes = n_classes,
    encoder = baseline_vgg16_encoder,
    image_augmenter = baseline_vgg16_imgaug,
    name = "baseline_vgg16_model",
    learning_rate = 0.001)
```

## 16.3 Training

```
[283]: train_evaluate_and_save_models(
    model = baseline_vgg16_model,
    encoder = baseline_vgg16_encoder,
    train_dataset = train_dataset,
    test_dataset = test_dataset,
    epochs = n_epochs,
    savename = "baseline_vgg16",
)
```

Training ...

Epoch 1/20  
241/241 [=====] - 467s 2s/step - loss: 2.1863 -  
accuracy: 0.4098 - accuracy\_top-5: 0.7435 - val\_loss: 1.7301 - val\_accuracy:  
0.4761 - val\_accuracy\_top-5: 0.8473  
Epoch 2/20  
241/241 [=====] - 461s 2s/step - loss: 1.0717 -  
accuracy: 0.6546 - accuracy\_top-5: 0.9430 - val\_loss: 1.7708 - val\_accuracy:  
0.4849 - val\_accuracy\_top-5: 0.8447  
Epoch 3/20  
241/241 [=====] - 463s 2s/step - loss: 0.8569 -  
accuracy: 0.7181 - accuracy\_top-5: 0.9644 - val\_loss: 1.6621 - val\_accuracy:  
0.5210 - val\_accuracy\_top-5: 0.8716  
Epoch 4/20  
241/241 [=====] - 452s 2s/step - loss: 0.7524 -  
accuracy: 0.7493 - accuracy\_top-5: 0.9745 - val\_loss: 1.6586 - val\_accuracy:  
0.5304 - val\_accuracy\_top-5: 0.8765  
Epoch 5/20  
241/241 [=====] - 450s 2s/step - loss: 0.6838 -  
accuracy: 0.7725 - accuracy\_top-5: 0.9779 - val\_loss: 1.5562 - val\_accuracy:  
0.5538 - val\_accuracy\_top-5: 0.8925  
Epoch 6/20  
241/241 [=====] - 447s 2s/step - loss: 0.6366 -  
accuracy: 0.7853 - accuracy\_top-5: 0.9811 - val\_loss: 1.6383 - val\_accuracy:  
0.5439 - val\_accuracy\_top-5: 0.8835  
Epoch 7/20  
241/241 [=====] - 447s 2s/step - loss: 0.5936 -  
accuracy: 0.8004 - accuracy\_top-5: 0.9834 - val\_loss: 1.4210 - val\_accuracy:  
0.6021 - val\_accuracy\_top-5: 0.9044  
Epoch 8/20  
241/241 [=====] - 446s 2s/step - loss: 0.5733 -  
accuracy: 0.8069 - accuracy\_top-5: 0.9838 - val\_loss: 1.8094 - val\_accuracy:  
0.5357 - val\_accuracy\_top-5: 0.8726  
Epoch 9/20  
241/241 [=====] - 446s 2s/step - loss: 0.5563 -  
accuracy: 0.8102 - accuracy\_top-5: 0.9872 - val\_loss: 1.5401 - val\_accuracy:  
0.5926 - val\_accuracy\_top-5: 0.8949  
Epoch 10/20  
241/241 [=====] - 446s 2s/step - loss: 0.5342 -  
accuracy: 0.8195 - accuracy\_top-5: 0.9867 - val\_loss: 1.6373 - val\_accuracy:  
0.5656 - val\_accuracy\_top-5: 0.8908  
Epoch 11/20  
241/241 [=====] - 447s 2s/step - loss: 0.5080 -  
accuracy: 0.8254 - accuracy\_top-5: 0.9889 - val\_loss: 1.7488 - val\_accuracy:  
0.5582 - val\_accuracy\_top-5: 0.8791  
Epoch 12/20  
241/241 [=====] - 445s 2s/step - loss: 0.4891 -  
accuracy: 0.8313 - accuracy\_top-5: 0.9892 - val\_loss: 1.5182 - val\_accuracy:  
0.6040 - val\_accuracy\_top-5: 0.9066

```
Epoch 13/20
241/241 [=====] - 446s 2s/step - loss: 0.4820 -
accuracy: 0.8357 - accuracy_top-5: 0.9895 - val_loss: 1.5841 - val_accuracy:
0.5943 - val_accuracy_top-5: 0.8953
Epoch 14/20
241/241 [=====] - 466s 2s/step - loss: 0.4681 -
accuracy: 0.8390 - accuracy_top-5: 0.9892 - val_loss: 1.5904 - val_accuracy:
0.5931 - val_accuracy_top-5: 0.8921
Epoch 15/20
241/241 [=====] - 530s 2s/step - loss: 0.4552 -
accuracy: 0.8405 - accuracy_top-5: 0.9916 - val_loss: 1.5642 - val_accuracy:
0.6005 - val_accuracy_top-5: 0.8990
Epoch 16/20
241/241 [=====] - 505s 2s/step - loss: 0.4464 -
accuracy: 0.8471 - accuracy_top-5: 0.9912 - val_loss: 1.6770 - val_accuracy:
0.5757 - val_accuracy_top-5: 0.8961
Epoch 17/20
241/241 [=====] - 496s 2s/step - loss: 0.4359 -
accuracy: 0.8488 - accuracy_top-5: 0.9919 - val_loss: 1.6840 - val_accuracy:
0.5899 - val_accuracy_top-5: 0.8887
Epoch 18/20
241/241 [=====] - 506s 2s/step - loss: 0.4279 -
accuracy: 0.8504 - accuracy_top-5: 0.9921 - val_loss: 1.8029 - val_accuracy:
0.5652 - val_accuracy_top-5: 0.8913
Epoch 19/20
241/241 [=====] - 503s 2s/step - loss: 0.4221 -
accuracy: 0.8526 - accuracy_top-5: 0.9918 - val_loss: 1.7181 - val_accuracy:
0.5866 - val_accuracy_top-5: 0.8913
Epoch 20/20
241/241 [=====] - 514s 2s/step - loss: 0.4184 -
accuracy: 0.8545 - accuracy_top-5: 0.9924 - val_loss: 1.5603 - val_accuracy:
0.6053 - val_accuracy_top-5: 0.9035

Saving models ...

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet
to be built. `model.compile_metrics` will be empty until you train or evaluate
the model.

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet
to be built. `model.compile_metrics` will be empty until you train or evaluate
the model.

INFO:tensorflow:Assets written to:
models/baseline_vgg16/encoder.saved_model/assets

INFO:tensorflow:Assets written to:
models/baseline_vgg16/encoder.saved_model/assets

INFO:tensorflow:Assets written to: models/baseline_vgg16/main.saved_model/assets
```

```
INFO:tensorflow:Assets written to: models/baseline_vgg16/main.saved_model/assets
```

```
Saving training history data ...
```

Results:

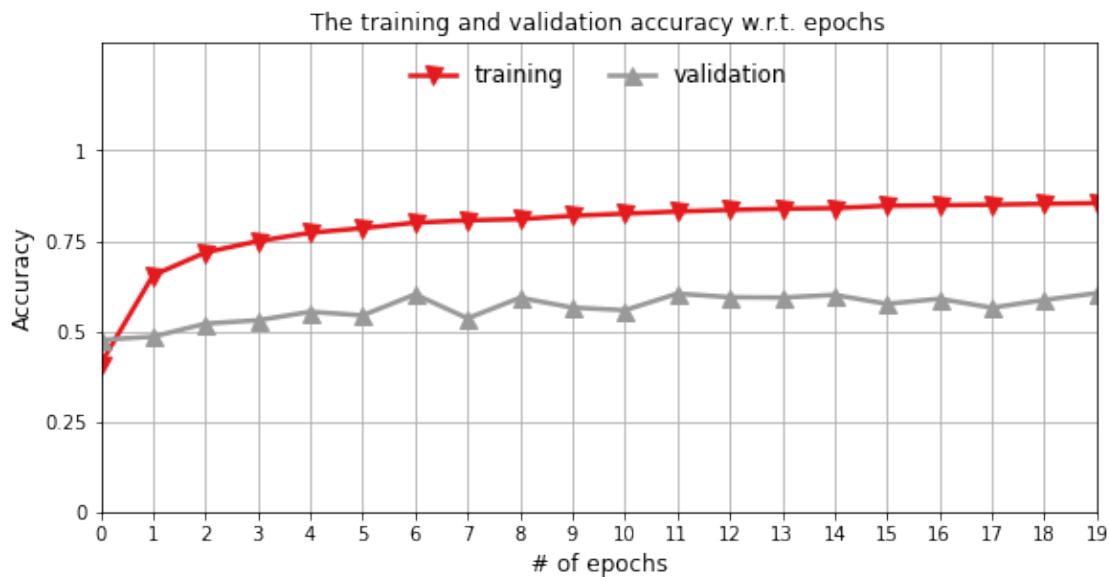
```
Maximal accuracy: 0.85448
Maximal accuracy_top-5: 0.99244
Maximal val_accuracy: 0.60532
Maximal val_accuracy_top-5: 0.90662
```

```
-- Completed --
```

## 16.4 Plotting results

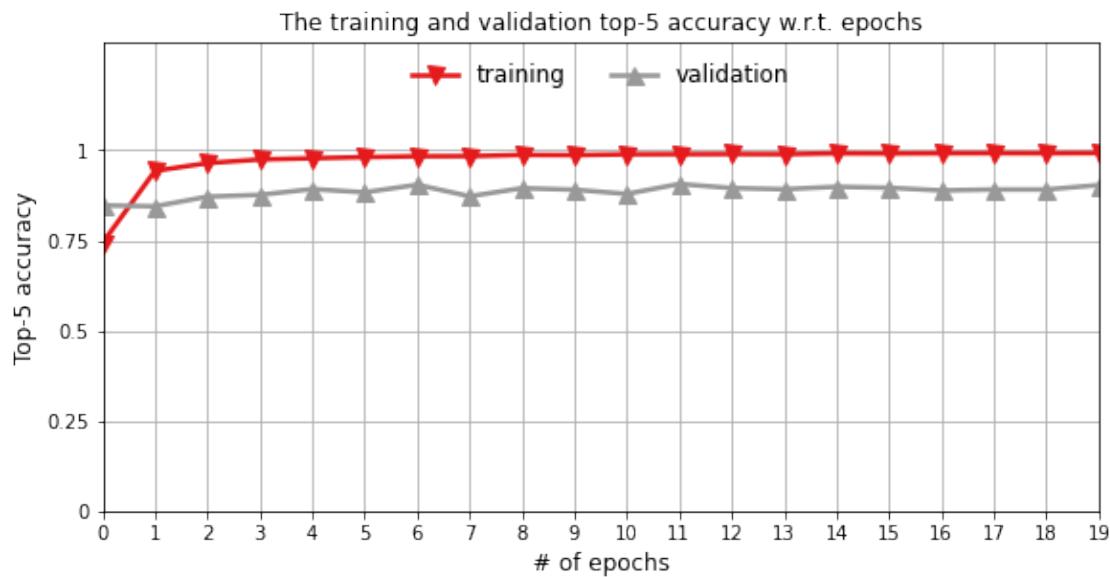
```
[18]: plot_train_results(
    resultfile = "results/baseline_vgg16/history_train.pickle",
    title = "The training and validation accuracy w.r.t. epochs",
    ylabel = "Accuracy",
    keys = ["accuracy", "val_accuracy"],
    labels = ["training", "validation"],
    savename = "fig_baseline_vgg16_accuracy.pdf",
    legend_location = "upper center",
    normalise = False
)
```

```
saved to: figures/fig_baseline_vgg16_accuracy.pdf
```



```
[19]: plot_train_results(
        resultfile = "results/baseline_vgg16/history_train.pickle",
        title = "The training and validation top-5 accuracy w.r.t. epochs",
        ylabel = "Top-5 accuracy",
        keys = ["accuracy_top-5", "val_accuracy_top-5"],
        labels = ["training", "validation"],
        savename = "fig_baseline_vgg16_accuracy_top5.pdf",
        legend_location = "upper center",
        normalise = False
    )
```

saved to: figures/fig\_baseline\_vgg16\_accuracy\_top5.pdf



## 17 Baseline: Supervised learning with ResNet50

### 17.1 Building ResNet50 encoder

```
[284]: from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input as resnet50_preprocess_input

def build_resnet50_encoder(image_shape):
```

- inputs = keras.Input(shape = image\_shape, name = "inputs")

```

#x = tf.keras.layers.Lambda(resnet50_preprocess_input)(inputs)

model_orig = ResNet50(
    include_top = False,      # no pre-defined head
    input_tensor = inputs,    # input tensor
    input_shape = image_shape, # input shape
    weights = "imagenet",
    pooling = "max"          #max avg
)

outputs = model_orig.layers[-1].output
#outputs = keras.layers.GlobalAveragePooling2D()(outputs)

model_new = keras.Model(inputs = inputs, outputs = outputs, name = "resnet50_encoder")

return model_new

```

## 17.2 Building model

```
[285]: baseline_resnet_imgaug = CustomisedTrainImageAugmenter(**imgaug_params)

baseline_resnet_encoder = build_resnet50_encoder(image_shape)
baseline_resnet_encoder.trainable = False

baseline_resnet_model = build_supervised_model(
    image_shape = image_shape,
    n_classes = n_classes,
    encoder = baseline_resnet_encoder,
    image_augmenter = baseline_resnet_imgaug,
    name = "baseline_resnet_model",
    learning_rate = 0.001)
```

## 17.3 Training

```
[286]: train_evaluate_and_save_models(
    model = baseline_resnet_model,
    encoder = baseline_resnet_encoder,
    train_dataset = train_dataset,
    test_dataset = test_dataset,
    epochs = n_epochs,
    savename = "baseline_resnet",
)
```

Training ...

Epoch 1/20  
241/241 [=====] - 233s 948ms/step - loss: 4.5451 -  
accuracy: 0.0314 - accuracy\_top-5: 0.1317 - val\_loss: 4.0374 - val\_accuracy:  
0.0523 - val\_accuracy\_top-5: 0.2010  
Epoch 2/20  
241/241 [=====] - 241s 996ms/step - loss: 3.8605 -  
accuracy: 0.0714 - accuracy\_top-5: 0.2698 - val\_loss: 4.0545 - val\_accuracy:  
0.0558 - val\_accuracy\_top-5: 0.2364  
Epoch 3/20  
241/241 [=====] - 227s 939ms/step - loss: 3.6530 -  
accuracy: 0.0949 - accuracy\_top-5: 0.3277 - val\_loss: 4.2026 - val\_accuracy:  
0.0596 - val\_accuracy\_top-5: 0.2452  
Epoch 4/20  
241/241 [=====] - 232s 963ms/step - loss: 3.5193 -  
accuracy: 0.1191 - accuracy\_top-5: 0.3691 - val\_loss: 4.7043 - val\_accuracy:  
0.0656 - val\_accuracy\_top-5: 0.2213  
Epoch 5/20  
241/241 [=====] - 208s 864ms/step - loss: 3.4487 -  
accuracy: 0.1279 - accuracy\_top-5: 0.3946 - val\_loss: 4.2191 - val\_accuracy:  
0.0732 - val\_accuracy\_top-5: 0.2803  
Epoch 6/20  
241/241 [=====] - 204s 845ms/step - loss: 3.3893 -  
accuracy: 0.1386 - accuracy\_top-5: 0.4148 - val\_loss: 4.2879 - val\_accuracy:  
0.0752 - val\_accuracy\_top-5: 0.2666  
Epoch 7/20  
241/241 [=====] - 204s 844ms/step - loss: 3.3272 -  
accuracy: 0.1454 - accuracy\_top-5: 0.4293 - val\_loss: 3.9732 - val\_accuracy:  
0.0991 - val\_accuracy\_top-5: 0.3201  
Epoch 8/20  
241/241 [=====] - 203s 843ms/step - loss: 3.2706 -  
accuracy: 0.1563 - accuracy\_top-5: 0.4495 - val\_loss: 4.1659 - val\_accuracy:  
0.1008 - val\_accuracy\_top-5: 0.3129  
Epoch 9/20  
241/241 [=====] - 203s 841ms/step - loss: 3.2218 -  
accuracy: 0.1636 - accuracy\_top-5: 0.4645 - val\_loss: 4.1969 - val\_accuracy:  
0.0882 - val\_accuracy\_top-5: 0.2995  
Epoch 10/20  
241/241 [=====] - 203s 843ms/step - loss: 3.1888 -  
accuracy: 0.1726 - accuracy\_top-5: 0.4691 - val\_loss: 4.2390 - val\_accuracy:  
0.0939 - val\_accuracy\_top-5: 0.3009  
Epoch 11/20  
241/241 [=====] - 211s 875ms/step - loss: 3.1338 -  
accuracy: 0.1761 - accuracy\_top-5: 0.4905 - val\_loss: 4.4350 - val\_accuracy:  
0.0887 - val\_accuracy\_top-5: 0.2860  
Epoch 12/20

```
241/241 [=====] - 207s 859ms/step - loss: 3.1264 -  
accuracy: 0.1821 - accuracy_top-5: 0.4937 - val_loss: 4.1547 - val_accuracy:  
0.1051 - val_accuracy_top-5: 0.3286  
Epoch 13/20  
241/241 [=====] - 208s 861ms/step - loss: 3.0884 -  
accuracy: 0.1872 - accuracy_top-5: 0.5026 - val_loss: 3.8290 - val_accuracy:  
0.1242 - val_accuracy_top-5: 0.3662  
Epoch 14/20  
241/241 [=====] - 208s 860ms/step - loss: 3.0680 -  
accuracy: 0.1914 - accuracy_top-5: 0.5090 - val_loss: 4.2247 - val_accuracy:  
0.1062 - val_accuracy_top-5: 0.3161  
Epoch 15/20  
241/241 [=====] - 208s 862ms/step - loss: 3.0076 -  
accuracy: 0.2031 - accuracy_top-5: 0.5265 - val_loss: 4.2707 - val_accuracy:  
0.1048 - val_accuracy_top-5: 0.3274  
Epoch 16/20  
241/241 [=====] - 208s 862ms/step - loss: 3.0037 -  
accuracy: 0.2019 - accuracy_top-5: 0.5287 - val_loss: 4.2068 - val_accuracy:  
0.1070 - val_accuracy_top-5: 0.3312  
Epoch 17/20  
241/241 [=====] - 208s 862ms/step - loss: 2.9690 -  
accuracy: 0.2067 - accuracy_top-5: 0.5383 - val_loss: 4.4619 - val_accuracy:  
0.1030 - val_accuracy_top-5: 0.3079  
Epoch 18/20  
241/241 [=====] - 208s 861ms/step - loss: 2.9677 -  
accuracy: 0.2117 - accuracy_top-5: 0.5359 - val_loss: 4.1042 - val_accuracy:  
0.1200 - val_accuracy_top-5: 0.3564  
Epoch 19/20  
241/241 [=====] - 207s 857ms/step - loss: 2.9301 -  
accuracy: 0.2207 - accuracy_top-5: 0.5469 - val_loss: 3.9254 - val_accuracy:  
0.1277 - val_accuracy_top-5: 0.3735  
Epoch 20/20  
241/241 [=====] - 208s 861ms/step - loss: 2.9103 -  
accuracy: 0.2210 - accuracy_top-5: 0.5523 - val_loss: 3.7140 - val_accuracy:  
0.1384 - val_accuracy_top-5: 0.3932
```

Saving models ...

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.

/usr/local/anaconda3/lib/python3.8/site-packages/tensorflow/python/keras/utils/generic\_utils.py:494: CustomMaskWarning: Custom mask layers require a config and must override get\_config. When loading,

```

the custom mask layer must be passed to the custom_objects argument.
    warnings.warn('Custom mask layers require a config and must override ')

INFO:tensorflow:Assets written to:
models/baseline_resnet/encoder.saved_model/assets

INFO:tensorflow:Assets written to:
models/baseline_resnet/encoder.saved_model/assets
/usr/local/anaconda3/lib/python3.8/site-
packages/tensorflow/python/keras/utils/generic_utils.py:494: CustomMaskWarning:
Custom mask layers require a config and must override get_config. When loading,
the custom mask layer must be passed to the custom_objects argument.
    warnings.warn('Custom mask layers require a config and must override ')

INFO:tensorflow:Assets written to:
models/baseline_resnet/main.saved_model/assets

INFO:tensorflow:Assets written to:
models/baseline_resnet/main.saved_model/assets

```

Saving training history data ...

Results:

```

Maximal accuracy: 0.22104
Maximal accuracy_top-5: 0.55231
Maximal val_accuracy: 0.13844
Maximal val_accuracy_top-5: 0.39325

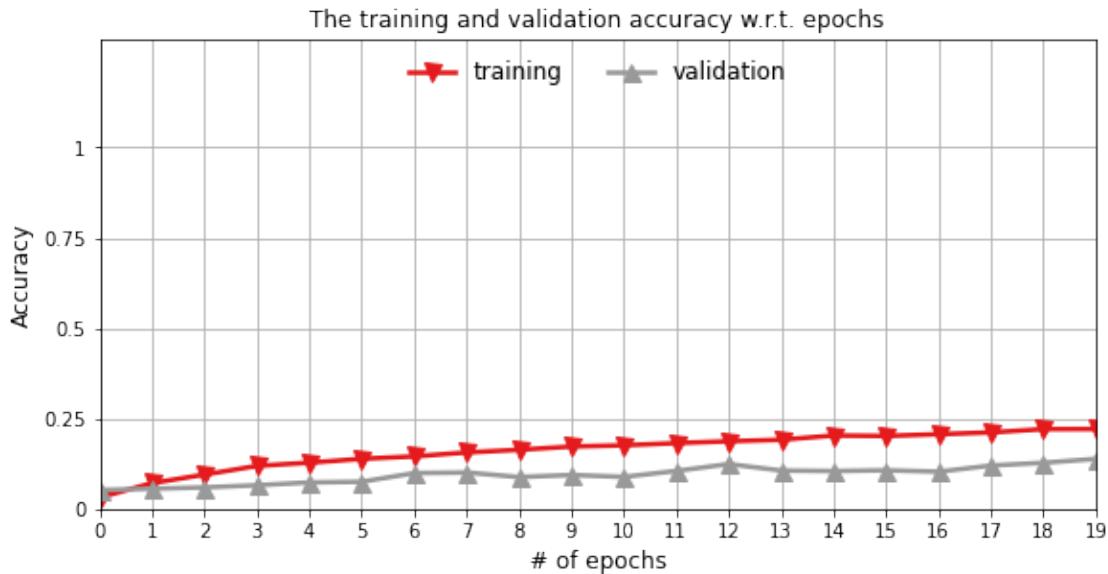
```

-- Completed --

## 17.4 Plotting results

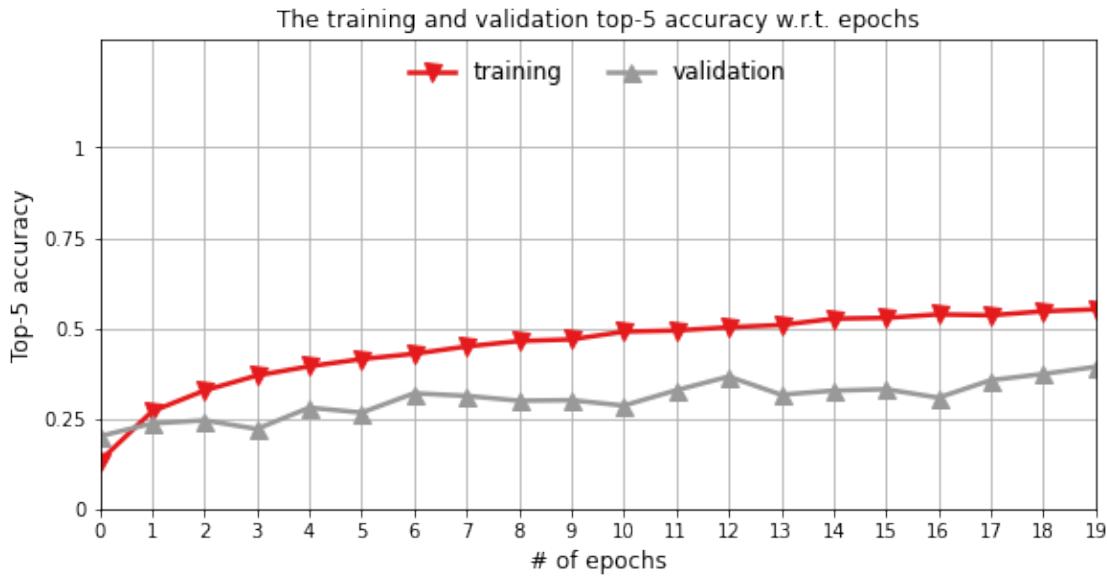
```
[20]: plot_train_results(
        resultfile = "results/baseline_resnet/history_train.pickle",
        title = "The training and validation accuracy w.r.t. epochs",
        ylabel = "Accuracy",
        keys = ["accuracy", "val_accuracy"],
        labels = ["training", "validation"],
        savename = "fig_baseline_resnet_accuracy.pdf",
        legend_location = "upper center",
        normalise = False
    )
```

saved to: figures/fig\_baseline\_resnet\_accuracy.pdf



```
[21]: plot_train_results(
        resultfile = "results/baseline_resnet/history_train.pickle",
        title = "The training and validation top-5 accuracy w.r.t.epochs",
        ylabel = "Top-5 accuracy",
        keys = ["accuracy_top-5", "val_accuracy_top-5"],
        labels = ["training", "validation"],
        savename = "fig_baseline_resnet_accuracy_top5.pdf",
        legend_location = "upper center",
        normalise = False
    )

saved to: figures/fig_baseline_resnet_accuracy_top5.pdf
```



## 18 Baseline: Supervised learning with MobileNet

### 18.1 Building MobileNet encoder

```
[287]: from tensorflow.keras.applications import MobileNet
from tensorflow.keras.applications.mobilenet import preprocess_input as u
↪mobilenet_preprocess_input

def build_mobilenet_encoder(image_shape):

    inputs = keras.Input(shape = image_shape, name = "inputs")

    #x = tf.keras.layers.Lambda(resnet50_preprocess_input)(inputs)

    model_orig = MobileNet(
        include_top = False,           # no pre-defined head
        input_tensor = inputs,         # input tensor
        input_shape = image_shape, # input shape
        weights = "imagenet",
        pooling = "max"                #max avg
    )

    outputs = model_orig.layers[-1].output
    #outputs = keras.layers.GlobalAveragePooling2D()(outputs)
```

```

model_new = keras.Model(inputs = inputs, outputs = outputs, name = "mobilenet_encoder")

return model_new

```

## 18.2 Building model

```
[288]: baseline_mobilenet_imgaug = CustomisedTrainImageAugmenter(**imgaug_params)

baseline_mobilenet_encoder = build_mobilenet_encoder(image_shape)
baseline_mobilenet_encoder.trainable = False

baseline_mobilenet_model = build_supervised_model(
    image_shape = image_shape,
    n_classes = n_classes,
    encoder = baseline_mobilenet_encoder,
    image_augmenter = baseline_mobilenet_imgaug,
    name = "baseline_mobilenet_model",
    learning_rate = 0.001)
```

WARNING:tensorflow: `input\_shape` is undefined or non-square, or `rows` is not in [128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.

WARNING:tensorflow: `input\_shape` is undefined or non-square, or `rows` is not in [128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.

## 18.3 Training

```
[289]: train_evaluate_and_save_models(
    model = baseline_mobilenet_model,
    encoder = baseline_mobilenet_encoder,
    train_dataset = train_dataset,
    test_dataset = test_dataset,
    epochs = n_epochs,
    savename = "baseline_mobilenet",
)
```

Training ...

Epoch 1/20  
241/241 [=====] - 64s 260ms/step - loss: 2.4931 -  
accuracy: 0.3477 - accuracy\_top-5: 0.6658 - val\_loss: 2.2340 - val\_accuracy:  
0.3710 - val\_accuracy\_top-5: 0.7335

Epoch 2/20  
241/241 [=====] - 67s 279ms/step - loss: 1.7051 -  
accuracy: 0.4984 - accuracy\_top-5: 0.8263 - val\_loss: 2.1422 - val\_accuracy:  
0.4055 - val\_accuracy\_top-5: 0.7561  
Epoch 3/20  
241/241 [=====] - 66s 272ms/step - loss: 1.5596 -  
accuracy: 0.5334 - accuracy\_top-5: 0.8530 - val\_loss: 2.1210 - val\_accuracy:  
0.4112 - val\_accuracy\_top-5: 0.7708  
Epoch 4/20  
241/241 [=====] - 66s 272ms/step - loss: 1.4793 -  
accuracy: 0.5541 - accuracy\_top-5: 0.8654 - val\_loss: 2.0638 - val\_accuracy:  
0.4256 - val\_accuracy\_top-5: 0.7782  
Epoch 5/20  
241/241 [=====] - 66s 273ms/step - loss: 1.4380 -  
accuracy: 0.5682 - accuracy\_top-5: 0.8699 - val\_loss: 2.0302 - val\_accuracy:  
0.4339 - val\_accuracy\_top-5: 0.7845  
Epoch 6/20  
241/241 [=====] - 66s 274ms/step - loss: 1.3908 -  
accuracy: 0.5783 - accuracy\_top-5: 0.8769 - val\_loss: 2.0262 - val\_accuracy:  
0.4456 - val\_accuracy\_top-5: 0.7845  
Epoch 7/20  
241/241 [=====] - 66s 273ms/step - loss: 1.3625 -  
accuracy: 0.5875 - accuracy\_top-5: 0.8826 - val\_loss: 2.0048 - val\_accuracy:  
0.4435 - val\_accuracy\_top-5: 0.7906  
Epoch 8/20  
241/241 [=====] - 66s 274ms/step - loss: 1.3366 -  
accuracy: 0.5960 - accuracy\_top-5: 0.8870 - val\_loss: 2.0435 - val\_accuracy:  
0.4421 - val\_accuracy\_top-5: 0.7814  
Epoch 9/20  
241/241 [=====] - 66s 275ms/step - loss: 1.3171 -  
accuracy: 0.6019 - accuracy\_top-5: 0.8903 - val\_loss: 2.0178 - val\_accuracy:  
0.4549 - val\_accuracy\_top-5: 0.7857  
Epoch 10/20  
241/241 [=====] - 66s 274ms/step - loss: 1.3008 -  
accuracy: 0.6036 - accuracy\_top-5: 0.8916 - val\_loss: 2.0072 - val\_accuracy:  
0.4512 - val\_accuracy\_top-5: 0.7903  
Epoch 11/20  
241/241 [=====] - 68s 281ms/step - loss: 1.2757 -  
accuracy: 0.6127 - accuracy\_top-5: 0.8944 - val\_loss: 2.0162 - val\_accuracy:  
0.4497 - val\_accuracy\_top-5: 0.7887  
Epoch 12/20  
241/241 [=====] - 66s 274ms/step - loss: 1.2496 -  
accuracy: 0.6158 - accuracy\_top-5: 0.8979 - val\_loss: 2.0461 - val\_accuracy:  
0.4540 - val\_accuracy\_top-5: 0.7899  
Epoch 13/20  
241/241 [=====] - 66s 275ms/step - loss: 1.2539 -  
accuracy: 0.6178 - accuracy\_top-5: 0.8978 - val\_loss: 1.9397 - val\_accuracy:  
0.4652 - val\_accuracy\_top-5: 0.8047

```
Epoch 14/20
241/241 [=====] - 66s 273ms/step - loss: 1.2385 -
accuracy: 0.6202 - accuracy_top-5: 0.8990 - val_loss: 2.0581 - val_accuracy:
0.4517 - val_accuracy_top-5: 0.7908
Epoch 15/20
241/241 [=====] - 67s 275ms/step - loss: 1.2247 -
accuracy: 0.6275 - accuracy_top-5: 0.9018 - val_loss: 2.0957 - val_accuracy:
0.4451 - val_accuracy_top-5: 0.7912
Epoch 16/20
241/241 [=====] - 66s 273ms/step - loss: 1.2138 -
accuracy: 0.6279 - accuracy_top-5: 0.9015 - val_loss: 1.9720 - val_accuracy:
0.4701 - val_accuracy_top-5: 0.7994
Epoch 17/20
241/241 [=====] - 66s 274ms/step - loss: 1.2114 -
accuracy: 0.6291 - accuracy_top-5: 0.9032 - val_loss: 1.9953 - val_accuracy:
0.4613 - val_accuracy_top-5: 0.7975
Epoch 18/20
241/241 [=====] - 67s 276ms/step - loss: 1.1877 -
accuracy: 0.6376 - accuracy_top-5: 0.9061 - val_loss: 2.0240 - val_accuracy:
0.4573 - val_accuracy_top-5: 0.7988
Epoch 19/20
241/241 [=====] - 66s 274ms/step - loss: 1.1915 -
accuracy: 0.6343 - accuracy_top-5: 0.9065 - val_loss: 2.0109 - val_accuracy:
0.4590 - val_accuracy_top-5: 0.8014
Epoch 20/20
241/241 [=====] - 67s 276ms/step - loss: 1.1861 -
accuracy: 0.6356 - accuracy_top-5: 0.9040 - val_loss: 2.0127 - val_accuracy:
0.4643 - val_accuracy_top-5: 0.7986
```

Saving models ...

```
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet
to be built. `model.compile_metrics` will be empty until you train or evaluate
the model.
```

```
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet
to be built. `model.compile_metrics` will be empty until you train or evaluate
the model.
```

```
INFO:tensorflow:Assets written to:
models/baseline_mobilenet/encoder.saved_model/assets
```

```
INFO:tensorflow:Assets written to:
models/baseline_mobilenet/encoder.saved_model/assets
```

```
INFO:tensorflow:Assets written to:
models/baseline_mobilenet/main.saved_model/assets
```

```
INFO:tensorflow:Assets written to:
models/baseline_mobilenet/main.saved_model/assets
```

```
Saving training history data ...
```

Results:

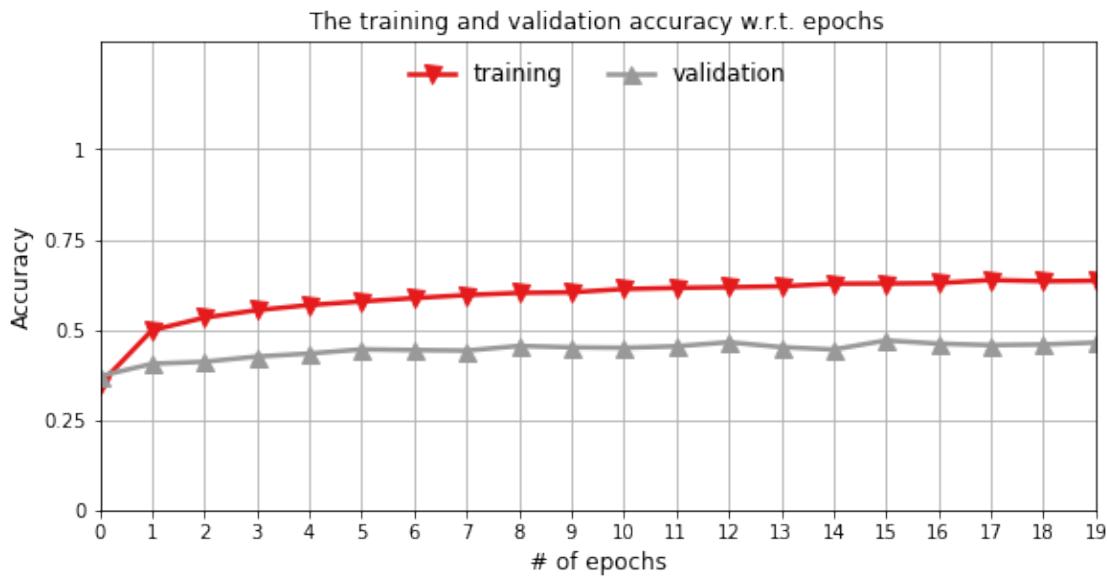
```
Maximal accuracy: 0.63760
Maximal accuracy_top-5: 0.90649
Maximal val_accuracy: 0.47013
Maximal val_accuracy_top-5: 0.80468
```

```
-- Completed --
```

## 18.4 Plotting results

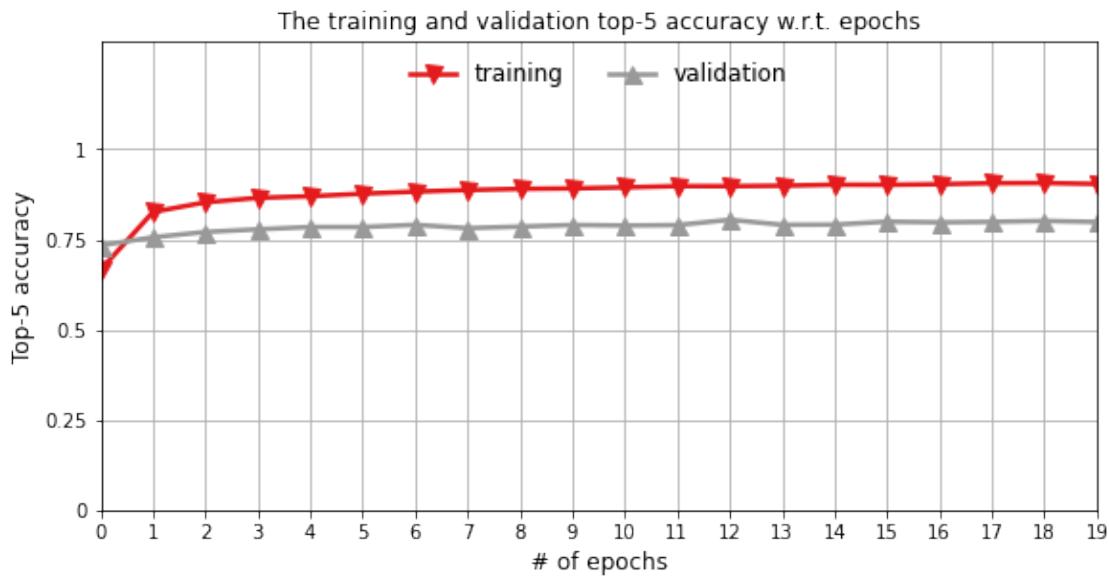
```
[22]: plot_train_results(
    resultfile = "results/baseline_mobilenet/history_train.pickle",
    title = "The training and validation accuracy w.r.t. epochs",
    ylabel = "Accuracy",
    keys = ["accuracy", "val_accuracy"],
    labels = ["training", "validation"],
    savename = "fig_baseline_mobilenet_accuracy.pdf",
    legend_location = "upper center",
    normalise = False
)
```

saved to: figures/fig\_baseline\_mobilenet\_accuracy.pdf



```
[23]: plot_train_results(
    resultfile = "results/baseline_mobilenet/history_train.pickle",
    title = "The training and validation top-5 accuracy w.r.t. ↵
    ↵epochs",
    ylabel = "Top-5 accuracy",
    keys = ["accuracy_top-5", "val_accuracy_top-5"],
    labels = ["training", "validation"],
    savename = "fig_baseline_mobilenet_accuracy_top5.pdf",
    legend_location = "upper center",
    normalise = False
)
```

saved to: figures/fig\_baseline\_mobilenet\_accuracy\_top5.pdf



## 19 Contrastive learning model

```
[17]: #We refer the framework from https://keras.io/examples/vision/
      ↵semisupervised_simclr/
```

```
import numpy as np

import tensorflow as tf
from tensorflow import keras

class ContrastiveModel(keras.Model):
```

```

def __init__(self,
            *,
            temperature = 0.5, #initial
            contrastive_loss_func = None, #use default loss
            encoder = None,
            projhead_hidden_dims = (512, 512,), #one hidden layer by default
            projhead_hidden_activation = "swish",
            projhead_output_dim = 512, #one linear output layer
            image_augmenter,
            n_classes, #number of classes, used for linear-regression metric
            **kwargs):
    #contrastive_loss_name_to_func_dict = {
    #    "infonce": self.infonce_contrastive_loss_func,
    #
    #    # "npairs": self.npairs_contrastive_loss_func
    #}
    #
    #assert (contrastive_loss_func in contrastive_loss_name_to_func_dict),
    #       "invalid contrastive_loss_func"
    #
    super(ContrastiveModel, self).__init__(**kwargs)

    self.initial_temperature = temperature
    self.current_temperature = self.initial_temperature

    self.image_augmenter = image_augmenter

    self.encoder = encoder

    self.n_classes = n_classes

    self.projhead_input_shape = self.encoder.layers[-1].output.shape[1]
    self.projhead_hidden_dims = projhead_hidden_dims
    self.projhead_hidden_activation = projhead_hidden_activation
    self.projhead_output_dim = projhead_output_dim

```

```

#print("projhead_input_shape = ", self.projhead_input_shape)
#print("projhead_hidden_dims = ", self.projhead_hidden_dims)
#print("projhead_hidden_activation = ", self.projhead_hidden_activation)
#print("projhead_output_dim = ", self.projhead_output_dim)

# build projection head
self.projhead_model = self.build_projhead_model(
    input_shape = self.projhead_input_shape,
    hidden_dims = self.projhead_hidden_dims,
    output_dim = self.projhead_output_dim
)

# Single dense layer for linear probing
self.probe_model = self.build_probe_model(
    n_classes = self.n_classes,
    input_shape = self.projhead_output_dim
)

#self.encoder.summary()
#self.projhead_model.summary()
#self.probe_model.summary()

if not contrastive_loss_func:
    self.contrastive_loss_func = self.infonce_contrastive_loss_func
else:
    self.contrastive_loss_func = contrastive_loss_func

def build_probe_model(self, *, n_classes, input_shape):
    return keras.Sequential(
        [
            keras.layers.InputLayer(input_shape = input_shape),
            keras.layers.Dense(n_classes), #pure linear
        ],
        name="linear_probe"
    )

def build_projhead_model(self, *, input_shape, hidden_dims, output_dim):

    inputs = keras.layers.Input(shape = input_shape, name = "projhead_input")

```

```

x = keras.layers.Flatten(name = "projhead_flatten")(inputs)

if not self.projhead_hidden_activation:
    self.projhead_hidden_activation = "swish"

for idx, dim in enumerate(hidden_dims):
    x = keras.layers.Dense(dim,
                           activation = self.projhead_hidden_activation,
                           name = f"projhead_hidden_{idx+1}")(x)

outputs = keras.layers.Dense(output_dim)(x)

model = keras.Model(inputs = inputs, outputs = outputs, name = "projhead_output")

return model

def compile(self, contrastive_optimizer, probe_optimizer, **kwargs):
    super().compile(**kwargs)

    self.contrastive_optimizer = contrastive_optimizer
    self.probe_optimizer = probe_optimizer

    self.probe_loss = keras.losses.
    ↪SparseCategoricalCrossentropy(from_logits=True)

    self.contrastive_loss_tracker = keras.metrics.Mean(
        name="contrastive_loss"
    )
    self.contrastive_accuracy = keras.metrics.SparseCategoricalAccuracy(
        name="contrastive_accuracy"
    )
    self.contrastive_accuracy_top5 = keras.metrics.
    ↪SparseTopKCategoricalAccuracy(
        5,
        name="contrastive_accuracy_top-5"
    )

    self.probe_loss_tracker = keras.metrics.Mean(
        name="probe_loss"
    )
    self.probe_accuracy = keras.metrics.SparseCategoricalAccuracy(
        name="probe_accuracy"
    )

```

```

        )

self.probe_accuracy_top5 = keras.metrics.SparseTopKCategoricalAccuracy(
    5,
    name="probe_accuracy_top-5"
)

@property
def metrics(self):
    return [
        self.contrastive_loss_tracker, #contrastive loss

        self.contrastive_accuracy, #contrastive accuracy
        self.contrastive_accuracy_top5,

        self.probe_loss_tracker, #linear prober loss

        self.probe_accuracy, #linear prober accuracy
        self.probe_accuracy_top5,
    ]
}

def infonce_contrastive_loss_func(self, projections_1, projections_2):

    if self.current_temperature <= 0:
        self.current_temperature = self.initial_temperature
        print(f"Reset current_temperature to {self.initial_temperature}")

    # We first normalise the projections for the two batches.
    projections_1 = tf.math.l2_normalize(projections_1, axis=1)
    projections_2 = tf.math.l2_normalize(projections_2, axis=1)

    # Computing  $A * B^T$ , all similarities will be computed at this point.
    similarities = (
        tf.matmul(projections_1, projections_2, transpose_b=True) / self.
    ↪current_temperature
    )

    batch_size = tf.shape(projections_1)[0]

    fake_labels = tf.range(batch_size)

    # Update the projection head accuracy
    self.contrastive_accuracy.update_state(fake_labels, similarities)
    self.contrastive_accuracy.update_state(fake_labels, tf.
    ↪transpose(similarities))

```

```

    self.contrastive_accuracy_top5.update_state(fake_labels, similarities)
    self.contrastive_accuracy_top5.update_state(fake_labels, tf.
→transpose(similarities))

# Compute the loss
loss_1_2 = keras.losses.sparse_categorical_crossentropy(
    fake_labels, similarities, from_logits=True
)

loss_2_1 = keras.losses.sparse_categorical_crossentropy(
    fake_labels, tf.transpose(similarities), from_logits=True
)

return (loss_1_2 + loss_2_1) / 2

def train_step(self, data):

#unpacking the training images and labels (fake) in the batch
images,labels = data

"""
The contrastive learning episode
"""

# We randomly sample two transformation functions
# and use them to augment the input batch (N images).
augmented_images_1 = self.image_augmenter(images)
augmented_images_2 = self.image_augmenter(images)

# We now have 2*N augmented images

# tf.GradientTape is a basic facility in tensorflow, which implemented
# of recording operations for automatic differentiation..
#
# ref: https://www.tensorflow.org/api_docs/python/tf/GradientTape
# ref:

# Create a GradientTape instance to record the forward process
with tf.GradientTape() as tape:
    # Feeding augmented images (2*N) into the base encoder
    # to extract their features (2*N)
    features_1 = self.encoder(augmented_images_1)

```

```

        features_2 = self.encoder(augmented_images_2)

        # Feeding the features (2*N) into the projection head, which is
        # a MLP, the final layer is a pure linear layer.
        projections_1 = self.projhead_model(features_1)
        projections_2 = self.projhead_model(features_2)

        # We now have 2*N projections from the projection head
        # and compute the contrastive loss (info-NCE)
        contrastive_loss = self.contrastive_loss_func(projections_1,
→projections_2)

        # Computing the gradients w.r.t. every trainable variables by
→back-propogating the errors
        # the *gradients* has the results with the form dL / dW
        gradients = tape.gradient(
            contrastive_loss,
            self.encoder.trainable_weights + self.projhead_model.
→trainable_weights,
            )

        # Optimising the weights by the computed gradients.
        self.contrastive_optimizer.apply_gradients(
            zip(
                gradients,
                self.encoder.trainable_weights + self.projhead_model.
→trainable_weights,
                )
            )

        # Updating the loss history.
        self.contrastive_loss_tracker.update_state(contrastive_loss)

    """
    Updating the weights of the linear prediction model
    """

augmented_images_3 = self.image_augmenter(images)

    # Forward
    with tf.GradientTape() as tape:

        features = self.encoder(augmented_images_3)

        features_2 = self.projhead_model(features)

```

```

    class_logits = self.probe_model(features_2)

    probe_loss = self.probe_loss(labels, class_logits)

    # backwards, computing the gradients w.r.t. the linear probe
    gradients = tape.gradient(probe_loss, self.probe_model.
    ↪trainable_weights)

    # updating the weights.
    self.probe_optimizer.apply_gradients(
        zip(gradients, self.probe_model.trainable_weights)
    )

    # updating history.
    self.probe_loss_tracker.update_state(probe_loss)

    self.probe_accuracy.update_state(labels, class_logits)
    self.probe_accuracy_top5.update_state(labels, class_logits)

    return {m.name: m.result() for m in self.metrics}

# for validation
def test_step(self, data):

    images, labels = data

    # Update probe
    augmented_images = self.image_augmenter(images, training=False)

    features = self.encoder(augmented_images, training=False)
    features_2 = self.projhead_model(features, training=False)

    class_logits = self.probe_model(features_2, training=False)
    probe_loss = self.probe_loss(labels, class_logits)

    self.probe_loss_tracker.update_state(probe_loss)

    self.probe_accuracy.update_state(labels, class_logits)
    self.probe_accuracy_top5.update_state(labels, class_logits)

    # Update contrastive
    augmented_images_1 = self.image_augmenter(images)
    augmented_images_2 = self.image_augmenter(images)

```

```

        features_1 = self.encoder(augmented_images_1)
        features_2 = self.encoder(augmented_images_2)

        projections_1 = self.projhead_model(features_1)
        projections_2 = self.projhead_model(features_2)

#accuracy will be computed here
 $_ = \text{self.contrastive\_loss\_func}(\text{projections\_1}, \text{projections\_2})$ 

    return {m.name: m.result() for m in self.metrics[2:]}

#{@tf.function
def call(self, inputs, training = False, mask = None):

    #augmented_images = self.image_augmenter(inputs)
    #features = self.encoder(augmented_images)

    features = self.encoder(inputs)
    outputs = self.projhead_model(features)

    return outputs

```

## 20 Scheduling training temperature

We compute the training temperature at the k-th epoch by:

$$T_k = \alpha^k T_0$$

```
[18]: class ContrastiveTemperatureScheduler(keras.callbacks.Callback):
    def __init__(self,
                alpha = 0.9,
                stop_temperature = 0.05):

        super(ContrastiveTemperatureScheduler, self).__init__()

        self.alpha = alpha
        self.stop_temperature = stop_temperature

    def on_train_begin(self, logs=None):
        self.model.current_temperature = self.model.initial_temperature

    def on_train_end(self, logs=None):
        self.model.current_temperature = self.model.initial_temperature

    def on_epoch_begin(self, epoch, logs=None):
```

```

        self.model.current_temperature = self.model.current_temperature * self.
    ↪alpha

    if self.model.current_temperature < self.stop_temperature:
        self.model.current_temperature = self.stop_temperature

def on_epoch_end(self, epoch, logs=None):
    pass

```

## 21 Contrastive learning with simple encoder

### 21.1 Building model

```
[337]: contrastive_simple_encoder = build_simple_encoder(image_shape)
contrastive_simple_encoder.trainable = True

contrastive_simple_imgaug = CustomisedTrainImageAugmenter(**imgaug_params)

contrastive_simple_model = ContrastiveModel(
    temperature = temperature,
    encoder = contrastive_simple_encoder,
    projhead_hidden_dims = (512, 512, ),
    projhead_hidden_activation = "swish",
    projhead_output_dim = 512,
    image_augmenter = contrastive_simple_imgaug,
    n_classes = n_classes
)

contrastive_simple_model.compile(
    contrastive_optimizer = keras.optimizers.Adam(learning_rate = 0.
    ↪001),
    probe_optimizer = keras.optimizers.Adam(),
)
```

### 21.2 Training

```
[338]: train_evaluate_and_save_models(
    model = contrastive_simple_model,
    encoder = contrastive_simple_encoder,
    train_dataset = train_dataset,
    test_dataset = test_dataset,
    epochs = n_epochs,
    savename = "contrastive_simple",
```

```
)
```

Training ...

Epoch 1/20

```
241/241 [=====] - 87s 344ms/step - contrastive_loss: 2.2404 - contrastive_accuracy: 0.7355 - contrastive_accuracy_top-5: 0.8958 - probe_loss: 4.5939 - probe_accuracy: 0.0335 - probe_accuracy_top-5: 0.1385 - val_contrastive_accuracy_top-5: 0.9623 - val_probe_loss: 4.4781 - val_probe_accuracy: 0.0339 - val_probe_accuracy_top-5: 0.1449
```

Epoch 2/20

```
241/241 [=====] - 77s 319ms/step - contrastive_loss: 1.6832 - contrastive_accuracy: 0.8894 - contrastive_accuracy_top-5: 0.9808 - probe_loss: 3.9923 - probe_accuracy: 0.0760 - probe_accuracy_top-5: 0.2721 - val_contrastive_accuracy_top-5: 0.9908 - val_probe_loss: 4.2383 - val_probe_accuracy: 0.0439 - val_probe_accuracy_top-5: 0.2005
```

Epoch 3/20

```
241/241 [=====] - 78s 322ms/step - contrastive_loss: 1.4508 - contrastive_accuracy: 0.9327 - contrastive_accuracy_top-5: 0.9946 - probe_loss: 3.6740 - probe_accuracy: 0.1038 - probe_accuracy_top-5: 0.3514 - val_contrastive_accuracy_top-5: 0.9954 - val_probe_loss: 4.0537 - val_probe_accuracy: 0.0582 - val_probe_accuracy_top-5: 0.2443
```

Epoch 4/20

```
241/241 [=====] - 79s 325ms/step - contrastive_loss: 1.3428 - contrastive_accuracy: 0.9618 - contrastive_accuracy_top-5: 0.9978 - probe_loss: 3.3929 - probe_accuracy: 0.1372 - probe_accuracy_top-5: 0.4265 - val_contrastive_accuracy_top-5: 0.9988 - val_probe_loss: 4.0570 - val_probe_accuracy: 0.0699 - val_probe_accuracy_top-5: 0.2627
```

Epoch 5/20

```
241/241 [=====] - 78s 324ms/step - contrastive_loss: 1.2696 - contrastive_accuracy: 0.9735 - contrastive_accuracy_top-5: 0.9991 - probe_loss: 3.1508 - probe_accuracy: 0.1753 - probe_accuracy_top-5: 0.4956 - val_contrastive_accuracy_top-5: 0.9988 - val_probe_loss: 3.7342 - val_probe_accuracy: 0.1086 - val_probe_accuracy_top-5: 0.3519
```

Epoch 6/20

```
241/241 [=====] - 79s 325ms/step - contrastive_loss: 1.2192 - contrastive_accuracy: 0.9804 - contrastive_accuracy_top-5: 0.9993 - probe_loss: 2.9469 - probe_accuracy: 0.2098 - probe_accuracy_top-5: 0.5539 - val_contrastive_accuracy_top-5: 0.9994 - val_probe_loss: 4.6687 - val_probe_accuracy: 0.0861 - val_probe_accuracy_top-5: 0.2721
```

Epoch 7/20

```
241/241 [=====] - 79s 327ms/step - contrastive_loss: 1.1874 - contrastive_accuracy: 0.9838 - contrastive_accuracy_top-5: 0.9998 - probe_loss: 2.8156 - probe_accuracy: 0.2294 - probe_accuracy_top-5: 0.5909 - val_contrastive_accuracy_top-5: 0.9995 - val_probe_loss: 3.6150 - val_probe_accuracy: 0.1495 - val_probe_accuracy_top-5: 0.4217
```

Epoch 8/20

```
241/241 [=====] - 79s 327ms/step - contrastive_loss: 1.1598 - contrastive_accuracy: 0.9875 - contrastive_accuracy_top-5: 0.9997 - probe_loss: 2.7121 - probe_accuracy: 0.2464 - probe_accuracy_top-5: 0.6199 - val_contrastive_accuracy_top-5: 0.9998 - val_probe_loss: 3.4005 - val_probe_accuracy: 0.1591 - val_probe_accuracy_top-5: 0.4770
Epoch 9/20
241/241 [=====] - 79s 327ms/step - contrastive_loss: 1.1434 - contrastive_accuracy: 0.9880 - contrastive_accuracy_top-5: 0.9999 - probe_loss: 2.6453 - probe_accuracy: 0.2586 - probe_accuracy_top-5: 0.6387 - val_contrastive_accuracy_top-5: 0.9998 - val_probe_loss: 3.6535 - val_probe_accuracy: 0.1469 - val_probe_accuracy_top-5: 0.4395
Epoch 10/20
241/241 [=====] - 79s 329ms/step - contrastive_loss: 1.1223 - contrastive_accuracy: 0.9879 - contrastive_accuracy_top-5: 0.9998 - probe_loss: 2.5299 - probe_accuracy: 0.2781 - probe_accuracy_top-5: 0.6661 - val_contrastive_accuracy_top-5: 0.9997 - val_probe_loss: 3.5019 - val_probe_accuracy: 0.1631 - val_probe_accuracy_top-5: 0.4729
Epoch 11/20
241/241 [=====] - 81s 336ms/step - contrastive_loss: 1.1199 - contrastive_accuracy: 0.9891 - contrastive_accuracy_top-5: 0.9999 - probe_loss: 2.5478 - probe_accuracy: 0.2755 - probe_accuracy_top-5: 0.6650 - val_contrastive_accuracy_top-5: 0.9997 - val_probe_loss: 3.5174 - val_probe_accuracy: 0.1629 - val_probe_accuracy_top-5: 0.4736
Epoch 12/20
241/241 [=====] - 80s 330ms/step - contrastive_loss: 1.1056 - contrastive_accuracy: 0.9905 - contrastive_accuracy_top-5: 0.9999 - probe_loss: 2.5096 - probe_accuracy: 0.2819 - probe_accuracy_top-5: 0.6751 - val_contrastive_accuracy_top-5: 0.9998 - val_probe_loss: 3.4485 - val_probe_accuracy: 0.1690 - val_probe_accuracy_top-5: 0.4899
Epoch 13/20
241/241 [=====] - 80s 331ms/step - contrastive_loss: 1.0958 - contrastive_accuracy: 0.9906 - contrastive_accuracy_top-5: 0.9999 - probe_loss: 2.4322 - probe_accuracy: 0.2989 - probe_accuracy_top-5: 0.6976 - val_contrastive_accuracy_top-5: 0.9995 - val_probe_loss: 3.5370 - val_probe_accuracy: 0.1735 - val_probe_accuracy_top-5: 0.4803
Epoch 14/20
241/241 [=====] - 80s 330ms/step - contrastive_loss: 1.0829 - contrastive_accuracy: 0.9918 - contrastive_accuracy_top-5: 0.9999 - probe_loss: 2.4299 - probe_accuracy: 0.2979 - probe_accuracy_top-5: 0.6951 - val_contrastive_accuracy_top-5: 1.0000 - val_probe_loss: 3.4338 - val_probe_accuracy: 0.1752 - val_probe_accuracy_top-5: 0.5074
Epoch 15/20
241/241 [=====] - 80s 330ms/step - contrastive_loss: 1.0838 - contrastive_accuracy: 0.9917 - contrastive_accuracy_top-5: 0.9999 - probe_loss: 2.4151 - probe_accuracy: 0.3012 - probe_accuracy_top-5: 0.7006 - val_contrastive_accuracy_top-5: 1.0000 - val_probe_loss: 3.4068 - val_probe_accuracy: 0.1823 - val_probe_accuracy_top-5: 0.5039
Epoch 16/20
```

```
241/241 [=====] - 82s 339ms/step - contrastive_loss: 1.0737 - contrastive_accuracy: 0.9924 - contrastive_accuracy_top-5: 0.9999 - probe_loss: 2.3480 - probe_accuracy: 0.3161 - probe_accuracy_top-5: 0.7148 - val_contrastive_accuracy_top-5: 0.9999 - val_probe_loss: 4.1869 - val_probe_accuracy: 0.1508 - val_probe_accuracy_top-5: 0.4313
Epoch 17/20
241/241 [=====] - 83s 341ms/step - contrastive_loss: 1.0660 - contrastive_accuracy: 0.9931 - contrastive_accuracy_top-5: 0.9999 - probe_loss: 2.3215 - probe_accuracy: 0.3233 - probe_accuracy_top-5: 0.7244 - val_contrastive_accuracy_top-5: 1.0000 - val_probe_loss: 3.4944 - val_probe_accuracy: 0.1777 - val_probe_accuracy_top-5: 0.5058
Epoch 18/20
241/241 [=====] - 82s 337ms/step - contrastive_loss: 1.0578 - contrastive_accuracy: 0.9933 - contrastive_accuracy_top-5: 1.0000 - probe_loss: 2.2736 - probe_accuracy: 0.3320 - probe_accuracy_top-5: 0.7344 - val_contrastive_accuracy_top-5: 0.9997 - val_probe_loss: 3.7932 - val_probe_accuracy: 0.1661 - val_probe_accuracy_top-5: 0.4842
Epoch 19/20
241/241 [=====] - 86s 356ms/step - contrastive_loss: 1.0518 - contrastive_accuracy: 0.9940 - contrastive_accuracy_top-5: 0.9999 - probe_loss: 2.2730 - probe_accuracy: 0.3318 - probe_accuracy_top-5: 0.7407 - val_contrastive_accuracy_top-5: 0.9998 - val_probe_loss: 3.3576 - val_probe_accuracy: 0.1848 - val_probe_accuracy_top-5: 0.5235
Epoch 20/20
241/241 [=====] - 89s 366ms/step - contrastive_loss: 1.0388 - contrastive_accuracy: 0.9941 - contrastive_accuracy_top-5: 1.0000 - probe_loss: 2.1412 - probe_accuracy: 0.3546 - probe_accuracy_top-5: 0.7669 - val_contrastive_accuracy_top-5: 1.0000 - val_probe_loss: 3.4939 - val_probe_accuracy: 0.1935 - val_probe_accuracy_top-5: 0.5277
```

Saving models ...

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.

```
INFO:tensorflow:Assets written to:  
models/contrastive_simple/encoder.saved_model/assets
```

```
INFO:tensorflow:Assets written to:  
models/contrastive_simple/encoder.saved_model/assets  
WARNING:absl:Found untraced functions such as train_imgaug_layer_layer_call_fn,  
train_imgaug_layer_layer_call_and_return_conditional_losses,  
train_imgaug_layer_layer_call_fn,  
train_imgaug_layer_layer_call_and_return_conditional_losses,
```

```
train_imgaug_layer_layer_call_and_return_conditional_losses while saving  
(showing 5 of 5). These functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to:  
models/contrastive_simple/main.saved_model/assets  
INFO:tensorflow:Assets written to:  
models/contrastive_simple/main.saved_model/assets
```

```
Saving training history data ...
```

```
Results:
```

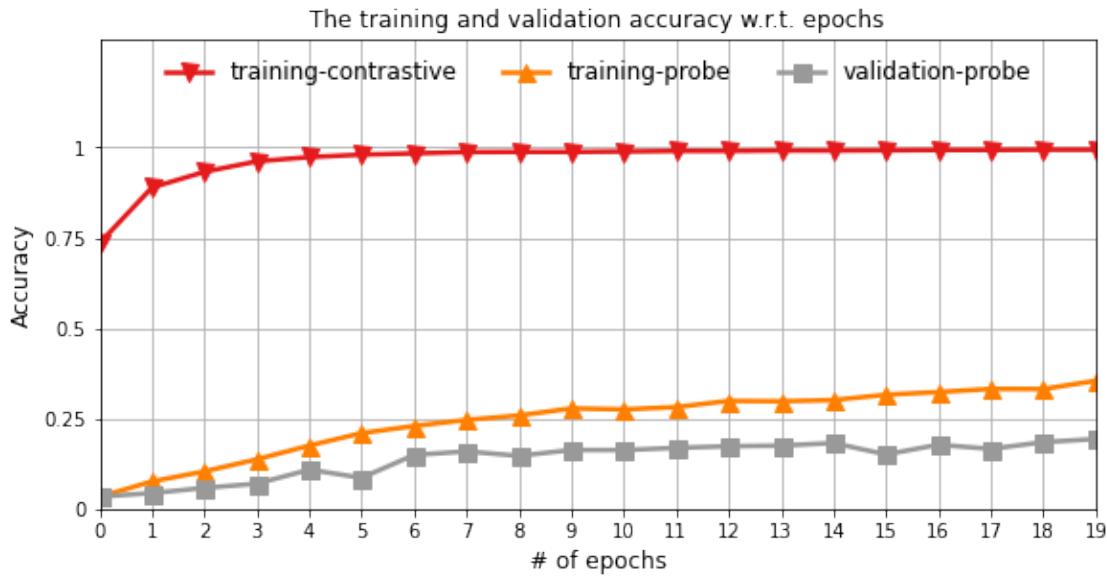
```
Maximal contrastive_accuracy: 0.99414  
Maximal contrastive_accuracy_top-5: 0.99997  
Maximal probe_accuracy: 0.35458  
Maximal probe_accuracy_top-5: 0.76692  
Maximal val_contrastive_accuracy_top-5: 1.00000  
Maximal val_probe_accuracy: 0.19351  
Maximal val_probe_accuracy_top-5: 0.52766
```

```
-- Completed --
```

### 21.3 Plotting results

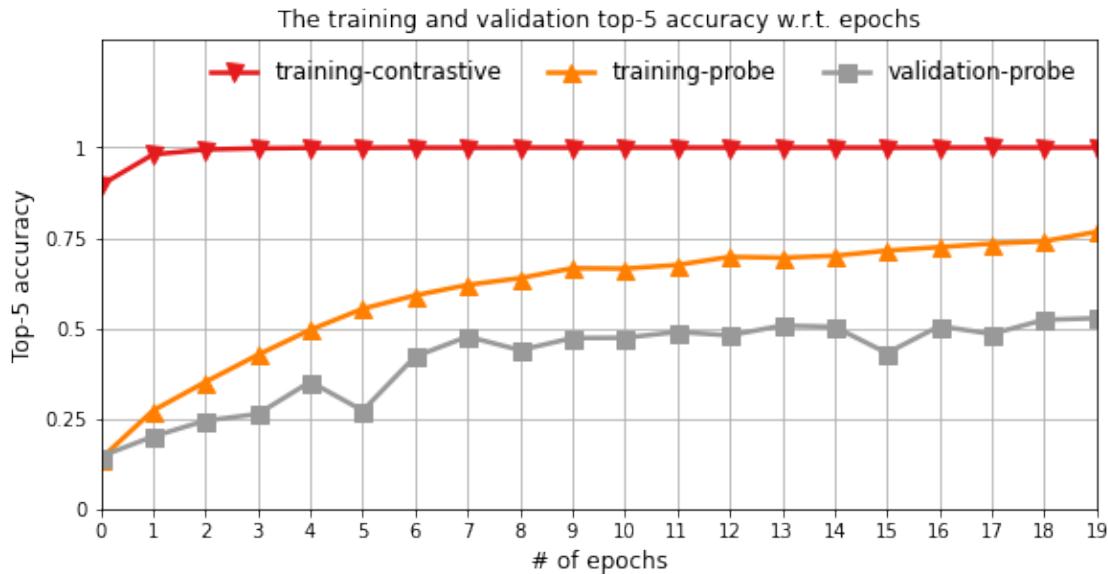
```
[24]: plot_train_results(  
        resultfile = "results/contrastive_simple/history_train.pickle",  
        title = "The training and validation accuracy w.r.t. epochs",  
        ylabel = "Accuracy",  
        keys = ["contrastive_accuracy",  
                "probe_accuracy", "val_probe_accuracy"],  
        labels = ["training-contrastive",  
                  "training-probe", "validation-probe"],  
        savename = "fig_contrastive_simple_accuracy.pdf",  
        legend_location = "upper center",  
        normalise = False  
)
```

saved to: figures/fig\_contrastive\_simple\_accuracy.pdf



```
[25]: plot_train_results(
        resultfile = "results/contrastive_simple/history_train.pickle",
        title = "The training and validation top-5 accuracy w.r.t.epochs",
        ylabel = "Top-5 accuracy",
        keys = ["contrastive_accuracy_top-5",
                "probe_accuracy_top-5", "val_probe_accuracy_top-5"],
        labels = ["training-contrastive",
                  "training-probe", "validation-probe"],
        savename = "fig_contrastive_simple_accuracy_top5.pdf",
        legend_location = "best",
        normalise = False
    )
```

saved to: figures/fig\_contrastive\_simple\_accuracy\_top5.pdf



## 22 Contrastive learning with simple encoder with dynamic temperature scheduling

```
[20]: contrastive_simple_dynamic_encoder = build_simple_encoder(image_shape)
contrastive_simple_dynamic_encoder.trainable = True

contrastive_simple_dynamic_imgaug = □
    ↳CustomisedTrainImageAugmenter(**imgaug_params)

contrastive_simple_dynamic_model = ContrastiveModel(
    temperature = temperature,
    encoder = contrastive_simple_dynamic_encoder,
    projhead_hidden_dims = (512, 512, ),
    projhead_hidden_activation = "swish",
    projhead_output_dim = 512,
    image_augmenter = □
        ↳contrastive_simple_dynamic_imgaug,
        n_classes = n_classes
    )

contrastive_simple_dynamic_model.compile(
    contrastive_optimizer = keras.optimizers.Adam(learning_rate = 0.
    ↳001),
```

```
        probe_optimizer = keras.optimizers.Adam(),
    )
```

## 22.1 Training

```
[22]: train_evaluate_and_save_models(
    model = contrastive_simple_dynamic_model,
    encoder = contrastive_simple_dynamic_encoder,
    train_dataset = train_dataset,
    test_dataset = test_dataset,
    epochs = n_epochs,
    savename = "contrastive_simple_dynamic",
    callbacks = [ContrastiveTemperatureScheduler()])
)
```

Training ...

```
Epoch 1/20
241/241 [=====] - 88s 353ms/step - contrastive_loss: 2.5335 - contrastive_accuracy: 0.5672 - contrastive_accuracy_top-5: 0.7870 - probe_loss: 4.4845 - probe_accuracy: 0.0468 - probe_accuracy_top-5: 0.1715 - val_contrastive_accuracy_top-5: 0.9185 - val_probe_loss: 4.4219 - val_probe_accuracy: 0.0356 - val_probe_accuracy_top-5: 0.1529
Epoch 2/20
241/241 [=====] - 91s 375ms/step - contrastive_loss: 1.6965 - contrastive_accuracy: 0.8186 - contrastive_accuracy_top-5: 0.9549 - probe_loss: 3.7177 - probe_accuracy: 0.1045 - probe_accuracy_top-5: 0.3474 - val_contrastive_accuracy_top-5: 0.9675 - val_probe_loss: 3.9528 - val_probe_accuracy: 0.0818 - val_probe_accuracy_top-5: 0.2892
Epoch 3/20
241/241 [=====] - 94s 388ms/step - contrastive_loss: 1.3693 - contrastive_accuracy: 0.8881 - contrastive_accuracy_top-5: 0.9825 - probe_loss: 3.3861 - probe_accuracy: 0.1538 - probe_accuracy_top-5: 0.4464 - val_contrastive_accuracy_top-5: 0.9884 - val_probe_loss: 3.8958 - val_probe_accuracy: 0.1169 - val_probe_accuracy_top-5: 0.3508
Epoch 4/20
241/241 [=====] - 95s 393ms/step - contrastive_loss: 1.2089 - contrastive_accuracy: 0.9300 - contrastive_accuracy_top-5: 0.9922 - probe_loss: 2.9352 - probe_accuracy: 0.2218 - probe_accuracy_top-5: 0.5703 - val_contrastive_accuracy_top-5: 0.9946 - val_probe_loss: 3.5694 - val_probe_accuracy: 0.1547 - val_probe_accuracy_top-5: 0.4336
Epoch 5/20
241/241 [=====] - 96s 395ms/step - contrastive_loss: 1.1042 - contrastive_accuracy: 0.9519 - contrastive_accuracy_top-5: 0.9969 - probe_loss: 2.6604 - probe_accuracy: 0.2773 - probe_accuracy_top-5: 0.6478 - val_contrastive_accuracy_top-5: 0.9978 - val_probe_loss: 3.3512 - val_probe_accuracy: 0.2069 - val_probe_accuracy_top-5: 0.5117
```

```
Epoch 6/20
241/241 [=====] - 92s 381ms/step - contrastive_loss: 1.0324 - contrastive_accuracy: 0.9635 - contrastive_accuracy_top-5: 0.9984 - probe_loss: 2.4073 - probe_accuracy: 0.3205 - probe_accuracy_top-5: 0.7140 - val_contrastive_accuracy_top-5: 0.9990 - val_probe_loss: 2.9717 - val_probe_accuracy: 0.2379 - val_probe_accuracy_top-5: 0.5909
Epoch 7/20
241/241 [=====] - 93s 386ms/step - contrastive_loss: 0.9723 - contrastive_accuracy: 0.9724 - contrastive_accuracy_top-5: 0.9993 - probe_loss: 2.0910 - probe_accuracy: 0.3924 - probe_accuracy_top-5: 0.7823 - val_contrastive_accuracy_top-5: 0.9985 - val_probe_loss: 3.9247 - val_probe_accuracy: 0.2009 - val_probe_accuracy_top-5: 0.4953
Epoch 8/20
241/241 [=====] - 94s 389ms/step - contrastive_loss: 0.9335 - contrastive_accuracy: 0.9759 - contrastive_accuracy_top-5: 0.9994 - probe_loss: 1.9094 - probe_accuracy: 0.4324 - probe_accuracy_top-5: 0.8205 - val_contrastive_accuracy_top-5: 0.9990 - val_probe_loss: 3.1374 - val_probe_accuracy: 0.2726 - val_probe_accuracy_top-5: 0.6123
Epoch 9/20
241/241 [=====] - 95s 392ms/step - contrastive_loss: 0.9013 - contrastive_accuracy: 0.9793 - contrastive_accuracy_top-5: 0.9997 - probe_loss: 1.7211 - probe_accuracy: 0.4787 - probe_accuracy_top-5: 0.8546 - val_contrastive_accuracy_top-5: 0.9996 - val_probe_loss: 2.5327 - val_probe_accuracy: 0.3422 - val_probe_accuracy_top-5: 0.7045
Epoch 10/20
241/241 [=====] - 96s 398ms/step - contrastive_loss: 0.8716 - contrastive_accuracy: 0.9828 - contrastive_accuracy_top-5: 0.9999 - probe_loss: 1.5852 - probe_accuracy: 0.5167 - probe_accuracy_top-5: 0.8749 - val_contrastive_accuracy_top-5: 0.9994 - val_probe_loss: 2.5986 - val_probe_accuracy: 0.3503 - val_probe_accuracy_top-5: 0.7014
Epoch 11/20
241/241 [=====] - 94s 387ms/step - contrastive_loss: 0.8451 - contrastive_accuracy: 0.9859 - contrastive_accuracy_top-5: 0.9999 - probe_loss: 1.5013 - probe_accuracy: 0.5396 - probe_accuracy_top-5: 0.8889 - val_contrastive_accuracy_top-5: 0.9999 - val_probe_loss: 2.3689 - val_probe_accuracy: 0.3977 - val_probe_accuracy_top-5: 0.7345
Epoch 12/20
241/241 [=====] - 94s 390ms/step - contrastive_loss: 0.8214 - contrastive_accuracy: 0.9868 - contrastive_accuracy_top-5: 0.9999 - probe_loss: 1.4268 - probe_accuracy: 0.5588 - probe_accuracy_top-5: 0.8999 - val_contrastive_accuracy_top-5: 0.9999 - val_probe_loss: 3.3171 - val_probe_accuracy: 0.3096 - val_probe_accuracy_top-5: 0.6308
Epoch 13/20
241/241 [=====] - 96s 399ms/step - contrastive_loss: 0.7974 - contrastive_accuracy: 0.9877 - contrastive_accuracy_top-5: 1.0000 - probe_loss: 1.2943 - probe_accuracy: 0.5976 - probe_accuracy_top-5: 0.9170 - val_contrastive_accuracy_top-5: 1.0000 - val_probe_loss: 2.3739 - val_probe_accuracy: 0.4209 - val_probe_accuracy_top-5: 0.7518
```

```
Epoch 14/20
241/241 [=====] - 96s 397ms/step - contrastive_loss: 0.7744 - contrastive_accuracy: 0.9901 - contrastive_accuracy_top-5: 1.0000 - probe_loss: 1.2105 - probe_accuracy: 0.6189 - probe_accuracy_top-5: 0.9295 - val_contrastive_accuracy_top-5: 1.0000 - val_probe_loss: 2.1653 - val_probe_accuracy: 0.4591 - val_probe_accuracy_top-5: 0.7795
Epoch 15/20
241/241 [=====] - 95s 395ms/step - contrastive_loss: 0.7616 - contrastive_accuracy: 0.9904 - contrastive_accuracy_top-5: 1.0000 - probe_loss: 1.1584 - probe_accuracy: 0.6323 - probe_accuracy_top-5: 0.9353 - val_contrastive_accuracy_top-5: 1.0000 - val_probe_loss: 2.7344 - val_probe_accuracy: 0.3947 - val_probe_accuracy_top-5: 0.7182
Epoch 16/20
241/241 [=====] - 95s 393ms/step - contrastive_loss: 0.7464 - contrastive_accuracy: 0.9912 - contrastive_accuracy_top-5: 1.0000 - probe_loss: 1.0680 - probe_accuracy: 0.6632 - probe_accuracy_top-5: 0.9451 - val_contrastive_accuracy_top-5: 1.0000 - val_probe_loss: 2.6500 - val_probe_accuracy: 0.4077 - val_probe_accuracy_top-5: 0.7431
Epoch 17/20
241/241 [=====] - 94s 388ms/step - contrastive_loss: 0.7349 - contrastive_accuracy: 0.9919 - contrastive_accuracy_top-5: 1.0000 - probe_loss: 1.0046 - probe_accuracy: 0.6769 - probe_accuracy_top-5: 0.9520 - val_contrastive_accuracy_top-5: 1.0000 - val_probe_loss: 2.2336 - val_probe_accuracy: 0.4765 - val_probe_accuracy_top-5: 0.7857
Epoch 18/20
241/241 [=====] - 95s 394ms/step - contrastive_loss: 0.7186 - contrastive_accuracy: 0.9921 - contrastive_accuracy_top-5: 1.0000 - probe_loss: 0.9369 - probe_accuracy: 0.6992 - probe_accuracy_top-5: 0.9566 - val_contrastive_accuracy_top-5: 1.0000 - val_probe_loss: 2.1590 - val_probe_accuracy: 0.5055 - val_probe_accuracy_top-5: 0.8118
Epoch 19/20
241/241 [=====] - 94s 388ms/step - contrastive_loss: 0.7092 - contrastive_accuracy: 0.9926 - contrastive_accuracy_top-5: 1.0000 - probe_loss: 0.8821 - probe_accuracy: 0.7181 - probe_accuracy_top-5: 0.9628 - val_contrastive_accuracy_top-5: 1.0000 - val_probe_loss: 1.9608 - val_probe_accuracy: 0.5317 - val_probe_accuracy_top-5: 0.8308
Epoch 20/20
241/241 [=====] - 90s 372ms/step - contrastive_loss: 0.6985 - contrastive_accuracy: 0.9932 - contrastive_accuracy_top-5: 1.0000 - probe_loss: 0.8508 - probe_accuracy: 0.7233 - probe_accuracy_top-5: 0.9651 - val_contrastive_accuracy_top-5: 1.0000 - val_probe_loss: 1.9943 - val_probe_accuracy: 0.5352 - val_probe_accuracy_top-5: 0.8242
```

Saving models ...

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.

```

INFO:tensorflow:Assets written to:
models/contrastive_simple_dynamic/encoder.saved_model/assets

WARNING:absl:Found untraced functions such as
train_imgaug_layer_layer_call_and_return_conditional_losses,
train_imgaug_layer_layer_call_fn, train_imgaug_layer_layer_call_fn,
train_imgaug_layer_layer_call_and_return_conditional_losses,
train_imgaug_layer_layer_call_and_return_conditional_losses while saving
(showing 5 of 5). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to:
models/contrastive_simple_dynamic/main.saved_model/assets

INFO:tensorflow:Assets written to:
models/contrastive_simple_dynamic/main.saved_model/assets

```

Saving training history data ...

Results:

```

Maximal contrastive_accuracy: 0.99317
Maximal contrastive_accuracy_top-5: 1.00000
Maximal probe_accuracy: 0.72334
Maximal probe_accuracy_top-5: 0.96506
Maximal val_contrastive_accuracy_top-5: 1.00000
Maximal val_probe_accuracy: 0.53519
Maximal val_probe_accuracy_top-5: 0.83078

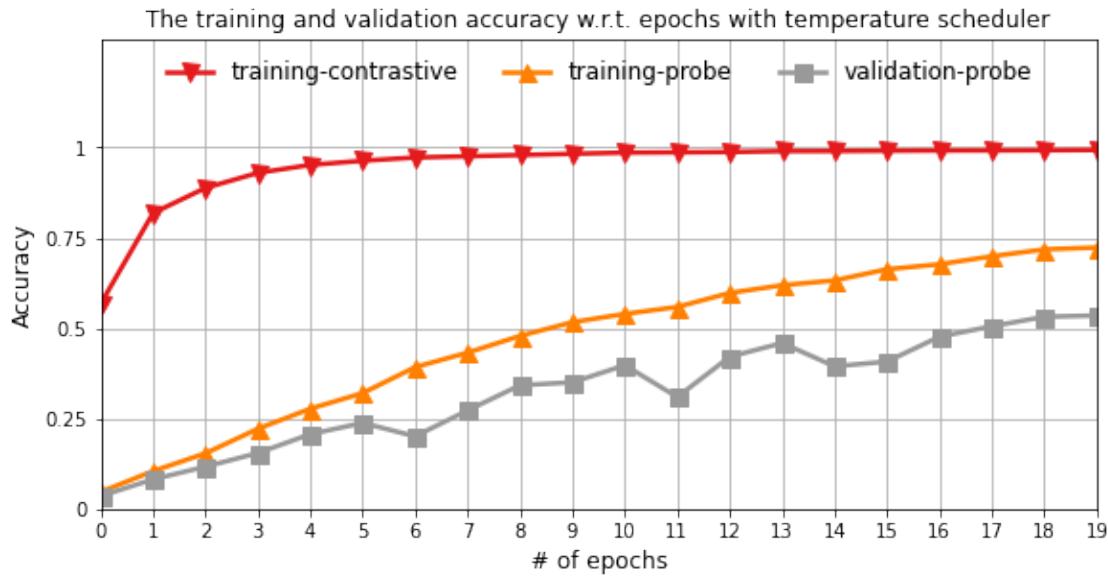
```

-- Completed --

## 22.2 Plotting results

```
[23]: plot_train_results(
            resultfile = "results/contrastive_simple_dynamic/history_train.
            →pickle",
            title = "The training and validation accuracy w.r.t. epochs
            →with temperature scheduler",
            ylabel = "Accuracy",
            keys = ["contrastive_accuracy",
                    "probe_accuracy", "val_probe_accuracy"],
            labels = ["training-contrastive",
                      "training-probe", "validation-probe"],
            savename = "fig_contrastive_simple_dynamic_accuracy.pdf",
            legend_location = "upper center",
            normalise = False
        )
```

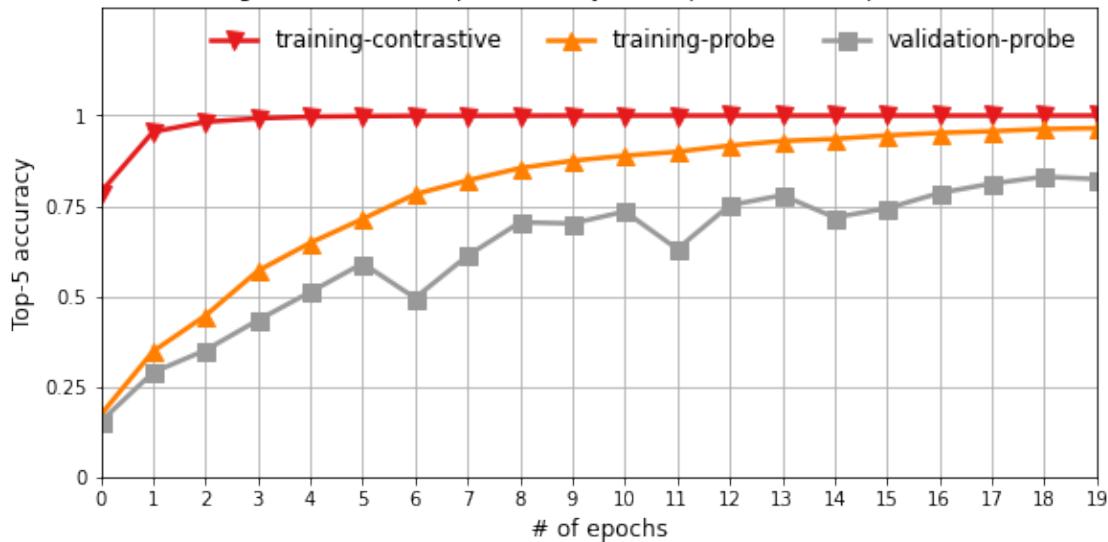
saved to: figures/fig\_contrastive\_simple\_dynamic\_accuracy.pdf



```
[24]: plot_train_results(
            resultfile = "results/contrastive_simple_dynamic/history_train.
            →pickle",
            title = "The training and validation top-5 accuracy w.r.t.↑
            →epochs with temperature scheduler",
            ylabel = "Top-5 accuracy",
            keys = ["contrastive_accuracy_top-5",
                    "probe_accuracy_top-5", "val_probe_accuracy_top-5"],
            labels = ["training-contrastive",
                      "training-probe", "validation-probe"],
            savename = "fig_contrastive_simple_dynamic_accuracy_top5.pdf",
            legend_location = "best",
            normalise = False
        )
```

saved to: figures/fig\_contrastive\_simple\_dynamic\_accuracy\_top5.pdf

The training and validation top-5 accuracy w.r.t. epochs with temperature scheduler



## 23 Contrastive learning with VGG16 without fine-tuning

```
[339]: contrastive_vgg16_encoder = build_vgg16_encoder(image_shape)
contrastive_vgg16_encoder.trainable = False

contrastive_vgg16_imgaug = CustomisedTrainImageAugmenter(**imgaug_params)

contrastive_vgg16_model = ContrastiveModel(
    temperature = temperature,
    encoder = contrastive_vgg16_encoder,
    projhead_hidden_dims = (512, 512, ),
    projhead_hidden_activation = "swish",
    projhead_output_dim = 512,
    image_augmenter = contrastive_vgg16_imgaug,
    n_classes = n_classes
)

contrastive_vgg16_model.compile(
    contrastive_optimizer = keras.optimizers.Adam(learning_rate = 0.
↪001),
    probe_optimizer = keras.optimizers.Adam(),
)
```

### 23.1 Training

```
[340]: train_evaluate_and_save_models(  
        model = contrastive_vgg16_model,  
        encoder = contrastive_vgg16_encoder,  
        train_dataset = train_dataset,  
        test_dataset = test_dataset,  
        epochs = n_epochs,  
        savename = "contrastive_vgg16",  
)
```

Training ...

Epoch 1/20

```
241/241 [=====] - 1194s 5s/step - contrastive_loss:  
1.7681 - contrastive_accuracy: 0.8889 - contrastive_accuracy_top-5: 0.9875 -  
probe_loss: 3.3692 - probe_accuracy: 0.4048 - probe_accuracy_top-5: 0.7501 -  
val_contrastive_accuracy_top-5: 0.9950 - val_probe_loss: 2.3002 -  
val_probe_accuracy: 0.4545 - val_probe_accuracy_top-5: 0.8314
```

Epoch 2/20

```
241/241 [=====] - 1204s 5s/step - contrastive_loss:  
1.5591 - contrastive_accuracy: 0.9196 - contrastive_accuracy_top-5: 0.9958 -  
probe_loss: 1.5817 - probe_accuracy: 0.6359 - probe_accuracy_top-5: 0.9307 -  
val_contrastive_accuracy_top-5: 0.9951 - val_probe_loss: 1.6658 -  
val_probe_accuracy: 0.5336 - val_probe_accuracy_top-5: 0.8696
```

Epoch 3/20

```
241/241 [=====] - 1199s 5s/step - contrastive_loss:  
1.4746 - contrastive_accuracy: 0.9192 - contrastive_accuracy_top-5: 0.9962 -  
probe_loss: 1.1043 - probe_accuracy: 0.6843 - probe_accuracy_top-5: 0.9516 -  
val_contrastive_accuracy_top-5: 0.9969 - val_probe_loss: 1.5470 -  
val_probe_accuracy: 0.5479 - val_probe_accuracy_top-5: 0.8803
```

Epoch 4/20

```
241/241 [=====] - 1262s 5s/step - contrastive_loss:  
1.4211 - contrastive_accuracy: 0.9216 - contrastive_accuracy_top-5: 0.9968 -  
probe_loss: 0.9430 - probe_accuracy: 0.7088 - probe_accuracy_top-5: 0.9582 -  
val_contrastive_accuracy_top-5: 0.9972 - val_probe_loss: 1.6620 -  
val_probe_accuracy: 0.5308 - val_probe_accuracy_top-5: 0.8732
```

Epoch 5/20

```
241/241 [=====] - 1501s 6s/step - contrastive_loss:  
1.3889 - contrastive_accuracy: 0.9225 - contrastive_accuracy_top-5: 0.9969 -  
probe_loss: 0.8625 - probe_accuracy: 0.7242 - probe_accuracy_top-5: 0.9646 -  
val_contrastive_accuracy_top-5: 0.9969 - val_probe_loss: 1.5644 -  
val_probe_accuracy: 0.5743 - val_probe_accuracy_top-5: 0.8913
```

Epoch 6/20

```
241/241 [=====] - 1559s 6s/step - contrastive_loss:  
1.3654 - contrastive_accuracy: 0.9225 - contrastive_accuracy_top-5: 0.9973 -  
probe_loss: 0.8213 - probe_accuracy: 0.7371 - probe_accuracy_top-5: 0.9692 -
```

```
val_contrastive_accuracy_top-5: 0.9973 - val_probe_loss: 1.6253 -
val_probe_accuracy: 0.5647 - val_probe_accuracy_top-5: 0.8873
Epoch 7/20
241/241 [=====] - 1458s 6s/step - contrastive_loss: 1.3493 - contrastive_accuracy: 0.9210 - contrastive_accuracy_top-5: 0.9970 - probe_loss: 0.8042 - probe_accuracy: 0.7412 - probe_accuracy_top-5: 0.9715 - val_contrastive_accuracy_top-5: 0.9979 - val_probe_loss: 1.7103 - val_probe_accuracy: 0.5701 - val_probe_accuracy_top-5: 0.8894
Epoch 8/20
241/241 [=====] - 1490s 6s/step - contrastive_loss: 1.3325 - contrastive_accuracy: 0.9228 - contrastive_accuracy_top-5: 0.9977 - probe_loss: 0.7767 - probe_accuracy: 0.7504 - probe_accuracy_top-5: 0.9733 - val_contrastive_accuracy_top-5: 0.9975 - val_probe_loss: 1.7753 - val_probe_accuracy: 0.5584 - val_probe_accuracy_top-5: 0.8886
Epoch 9/20
241/241 [=====] - 1521s 6s/step - contrastive_loss: 1.3150 - contrastive_accuracy: 0.9231 - contrastive_accuracy_top-5: 0.9976 - probe_loss: 0.7717 - probe_accuracy: 0.7547 - probe_accuracy_top-5: 0.9738 - val_contrastive_accuracy_top-5: 0.9971 - val_probe_loss: 1.8085 - val_probe_accuracy: 0.5701 - val_probe_accuracy_top-5: 0.8930
Epoch 10/20
241/241 [=====] - 1447s 6s/step - contrastive_loss: 1.3038 - contrastive_accuracy: 0.9233 - contrastive_accuracy_top-5: 0.9976 - probe_loss: 0.7564 - probe_accuracy: 0.7571 - probe_accuracy_top-5: 0.9759 - val_contrastive_accuracy_top-5: 0.9966 - val_probe_loss: 1.7845 - val_probe_accuracy: 0.5729 - val_probe_accuracy_top-5: 0.8943
Epoch 11/20
241/241 [=====] - 1407s 6s/step - contrastive_loss: 1.2898 - contrastive_accuracy: 0.9251 - contrastive_accuracy_top-5: 0.9979 - probe_loss: 0.7583 - probe_accuracy: 0.7574 - probe_accuracy_top-5: 0.9762 - val_contrastive_accuracy_top-5: 0.9958 - val_probe_loss: 1.7824 - val_probe_accuracy: 0.5823 - val_probe_accuracy_top-5: 0.8931
Epoch 12/20
241/241 [=====] - 1424s 6s/step - contrastive_loss: 1.2785 - contrastive_accuracy: 0.9245 - contrastive_accuracy_top-5: 0.9981 - probe_loss: 0.7580 - probe_accuracy: 0.7596 - probe_accuracy_top-5: 0.9762 - val_contrastive_accuracy_top-5: 0.9977 - val_probe_loss: 1.8548 - val_probe_accuracy: 0.5662 - val_probe_accuracy_top-5: 0.8875
Epoch 13/20
241/241 [=====] - 1383s 6s/step - contrastive_loss: 1.2749 - contrastive_accuracy: 0.9225 - contrastive_accuracy_top-5: 0.9981 - probe_loss: 0.7529 - probe_accuracy: 0.7588 - probe_accuracy_top-5: 0.9761 - val_contrastive_accuracy_top-5: 0.9968 - val_probe_loss: 1.8414 - val_probe_accuracy: 0.5722 - val_probe_accuracy_top-5: 0.8940
Epoch 14/20
241/241 [=====] - 1325s 6s/step - contrastive_loss: 1.2587 - contrastive_accuracy: 0.9278 - contrastive_accuracy_top-5: 0.9982 - probe_loss: 0.7193 - probe_accuracy: 0.7681 - probe_accuracy_top-5: 0.9780 -
```

```
val_contrastive_accuracy_top-5: 0.9966 - val_probe_loss: 1.9528 -
val_probe_accuracy: 0.5609 - val_probe_accuracy_top-5: 0.8912
Epoch 15/20
241/241 [=====] - 1340s 6s/step - contrastive_loss: 1.2500 - contrastive_accuracy: 0.9261 - contrastive_accuracy_top-5: 0.9979 - probe_loss: 0.7274 - probe_accuracy: 0.7674 - probe_accuracy_top-5: 0.9780 - val_contrastive_accuracy_top-5: 0.9955 - val_probe_loss: 1.9163 - val_probe_accuracy: 0.5690 - val_probe_accuracy_top-5: 0.8874
Epoch 16/20
241/241 [=====] - 1332s 6s/step - contrastive_loss: 1.2479 - contrastive_accuracy: 0.9238 - contrastive_accuracy_top-5: 0.9981 - probe_loss: 0.7316 - probe_accuracy: 0.7668 - probe_accuracy_top-5: 0.9775 - val_contrastive_accuracy_top-5: 0.9969 - val_probe_loss: 1.9087 - val_probe_accuracy: 0.5701 - val_probe_accuracy_top-5: 0.8870
Epoch 17/20
241/241 [=====] - 1253s 5s/step - contrastive_loss: 1.2400 - contrastive_accuracy: 0.9255 - contrastive_accuracy_top-5: 0.9980 - probe_loss: 0.7140 - probe_accuracy: 0.7733 - probe_accuracy_top-5: 0.9799 - val_contrastive_accuracy_top-5: 0.9972 - val_probe_loss: 1.9538 - val_probe_accuracy: 0.5694 - val_probe_accuracy_top-5: 0.8856
Epoch 18/20
241/241 [=====] - 1240s 5s/step - contrastive_loss: 1.2298 - contrastive_accuracy: 0.9275 - contrastive_accuracy_top-5: 0.9984 - probe_loss: 0.7101 - probe_accuracy: 0.7749 - probe_accuracy_top-5: 0.9788 - val_contrastive_accuracy_top-5: 0.9965 - val_probe_loss: 2.1624 - val_probe_accuracy: 0.5623 - val_probe_accuracy_top-5: 0.8790
Epoch 19/20
241/241 [=====] - 1232s 5s/step - contrastive_loss: 1.2245 - contrastive_accuracy: 0.9262 - contrastive_accuracy_top-5: 0.9984 - probe_loss: 0.7002 - probe_accuracy: 0.7770 - probe_accuracy_top-5: 0.9803 - val_contrastive_accuracy_top-5: 0.9955 - val_probe_loss: 2.1414 - val_probe_accuracy: 0.5505 - val_probe_accuracy_top-5: 0.8749
Epoch 20/20
241/241 [=====] - 1279s 5s/step - contrastive_loss: 1.2212 - contrastive_accuracy: 0.9243 - contrastive_accuracy_top-5: 0.9982 - probe_loss: 0.7031 - probe_accuracy: 0.7769 - probe_accuracy_top-5: 0.9802 - val_contrastive_accuracy_top-5: 0.9960 - val_probe_loss: 2.0835 - val_probe_accuracy: 0.5643 - val_probe_accuracy_top-5: 0.8799
```

Saving models ...

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.

```
INFO:tensorflow:Assets written to:  
models/contrastive_vgg16/encoder.saved_model/assets  
  
INFO:tensorflow:Assets written to:  
models/contrastive_vgg16/encoder.saved_model/assets  
WARNING:absl:Found untraced functions such as train_imgaug_layer_layer_call_fn,  
train_imgaug_layer_layer_call_and_return_conditional_losses,  
train_imgaug_layer_layer_call_fn,  
train_imgaug_layer_layer_call_and_return_conditional_losses,  
train_imgaug_layer_layer_call_and_return_conditional_losses while saving  
(showing 5 of 5). These functions will not be directly callable after loading.  
  
INFO:tensorflow:Assets written to:  
models/contrastive_vgg16/main.saved_model/assets  
  
INFO:tensorflow:Assets written to:  
models/contrastive_vgg16/main.saved_model/assets
```

Saving training history data ...

Results:

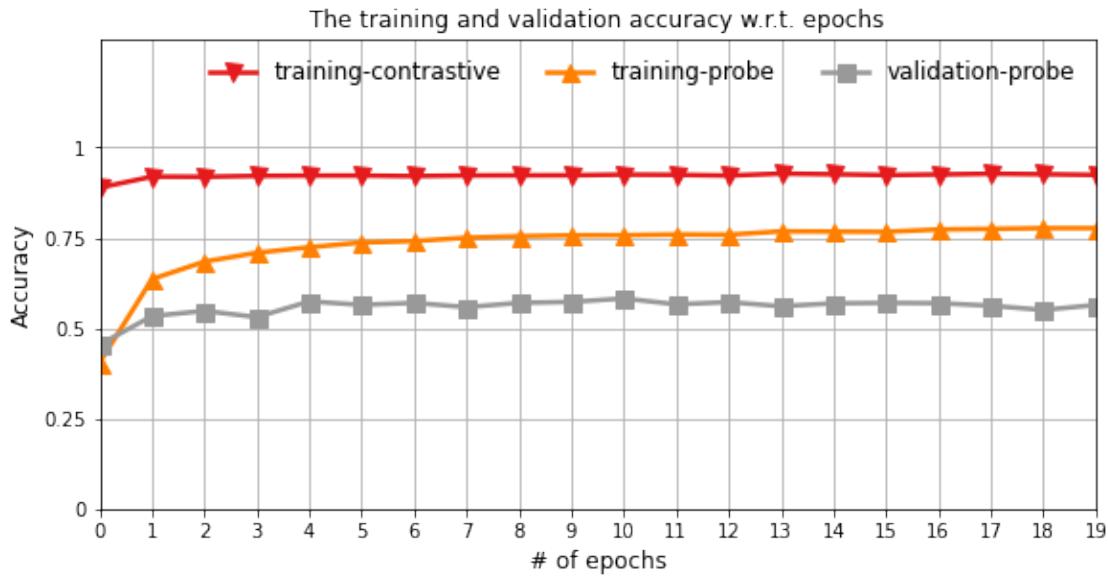
```
Maximal contrastive_accuracy: 0.92782  
Maximal contrastive_accuracy_top-5: 0.99838  
Maximal probe_accuracy: 0.77701  
Maximal probe_accuracy_top-5: 0.98029  
Maximal val_contrastive_accuracy_top-5: 0.99792  
Maximal val_probe_accuracy: 0.58234  
Maximal val_probe_accuracy_top-5: 0.89429
```

-- Completed --

## 23.2 Plotting results

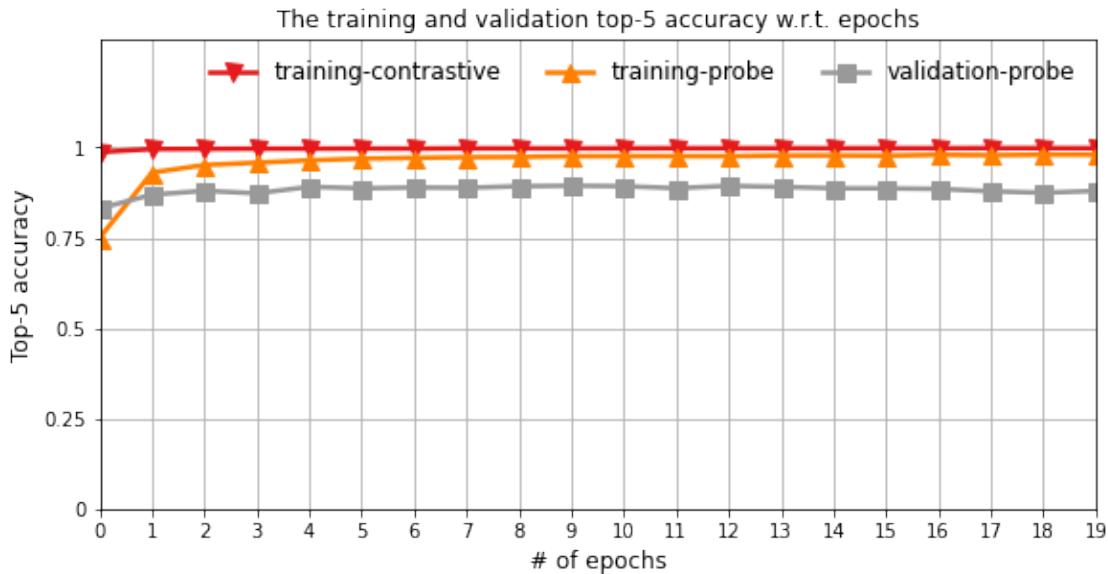
```
[26]: plot_train_results(  
        resultfile = "results/contrastive_vgg16/history_train.pickle",  
        title = "The training and validation accuracy w.r.t. epochs",  
        ylabel = "Accuracy",  
        keys = ["contrastive_accuracy",  
                "probe_accuracy", "val_probe_accuracy"],  
        labels = ["training-contrastive",  
                  "training-probe", "validation-probe"],  
        savename = "fig_contrastive_vgg16_accuracy.pdf",  
        legend_location = "best",  
        normalise = False  
)
```

saved to: figures/fig\_contrastive\_vgg16\_accuracy.pdf



```
[27]: plot_train_results(
    resultfile = "results/contrastive_vgg16/history_train.pickle",
    title = "The training and validation top-5 accuracy w.r.t. ↵
    ↵epochs",
    ylabel = "Top-5 accuracy",
    keys = ["contrastive_accuracy_top-5",
            "probe_accuracy_top-5", "val_probe_accuracy_top-5"],
    labels = ["training-contrastive",
              "training-probe", "validation-probe"],
    savename = "fig_contrastive_vgg16_accuracy_top5.pdf",
    legend_location = "best",
    normalise = False
)
```

saved to: figures/fig\_contrastive\_vgg16\_accuracy\_top5.pdf



## 24 Contrastive learning with vision transformer (ViT) encoder

### 24.1 Vision transformer (ViT) encoder

```
[ ]: # ref: https://keras.io/examples/vision/
    ↳image_classification_with_vision_transformer/
import tensorflow_addons as tfa

patch_size = 6
n_patches = (image_size[0] // patch_size) ** 2

patch_projection_dim = 64

#attention heads
n_heads = 4
```

### 24.2 Creating patches

```
[ ]: class Patches(keras.layers.Layer):
    def __init__(self, patch_size):
        super(Patches, self).__init__()
        self.patch_size = patch_size

    def call(self, images):
        batch_size = tf.shape(images)[0]
        patches = tf.image.extract_patches(
            images=images,
```

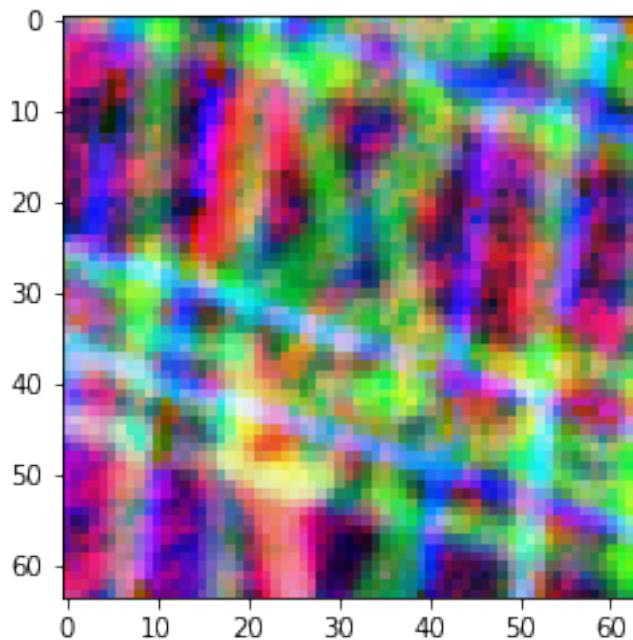
```

        sizes=[1, self.patch_size, self.patch_size, 1],
        strides=[1, self.patch_size, self.patch_size, 1],
        rates=[1, 1, 1, 1],
        padding="VALID",
    )
patch_dims = patches.shape[-1]
patches = tf.reshape(patches, [batch_size, -1, patch_dims])
return patches

```

```
[263]: x_batch = next(iter(train_dataset))[0]
x_batch = x_batch.numpy()
x_batch.shape

plt.imshow(x_batch[0])
plt.show()
```



```
[251]: import matplotlib.pyplot as plt

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

x_batch = next(iter(train_dataset))[0]
x_batch = x_batch.numpy()

image = x_batch[np.random.choice(range(x_batch.shape[0]))]
# plt.imshow(image.astype("uint8"))
```

```

plt.imshow(image)
plt.axis("off")

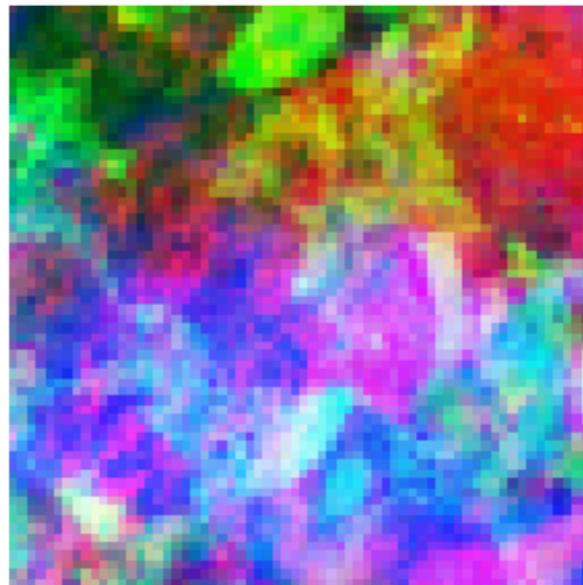
resized_image = tf.image.resize(
    tf.convert_to_tensor([image]), size=image_size
)

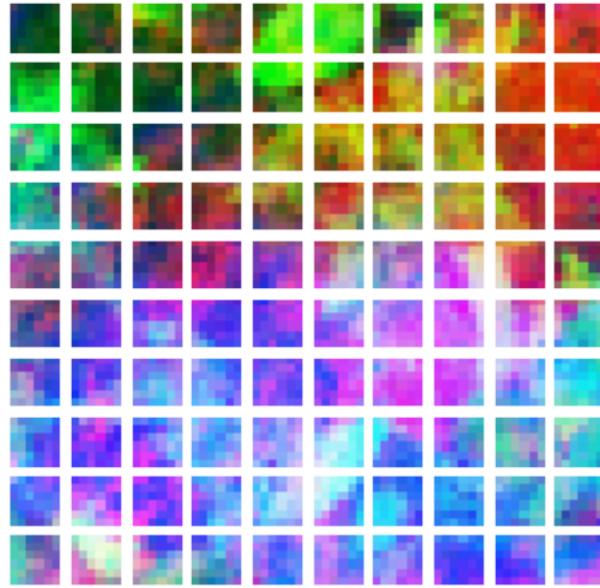
patches = Patches(patch_size)(resized_image)
print(f"Image size: {image_size}")
print(f"Patch size: {patch_size} X {patch_size}")
print(f"Patches per image: {patches.shape[1]}")
print(f"Elements per patch: {patches.shape[-1]}")

n = int(np.sqrt(patches.shape[1]))
plt.figure(figsize=(4, 4))
for i, patch in enumerate(patches[0]):
    ax = plt.subplot(n, n, i + 1)
    patch_img = tf.reshape(patch, (patch_size, patch_size, 3))
    #plt.imshow(patch_img.numpy().astype("uint8"))
    plt.imshow(patch_img.numpy())
    plt.axis("off")

```

Image size: (64, 64)  
 Patch size: 6 X 6  
 Patches per image: 100  
 Elements per patch: 108





### 24.3 Patch encoder

```
[ ]: class PatchEncoder(keras.layers.Layer):
    def __init__(self, n_patches, patch_projection_dim):
        super(PatchEncoder, self).__init__()
        self.n_patches = n_patches

        self.projection = keras.layers.Dense(units = patch_projection_dim)

        self.position_embedding = keras.layers.Embedding(
            input_dim = n_patches, output_dim=patch_projection_dim
        )

    def call(self, patch):
        positions = tf.range(start=0, limit=self.n_patches, delta=1)
        encoded = self.projection(patch) + self.position_embedding(positions)
        return encoded
```

### 24.4 ViT MLP

```
[ ]: def vit_mlp(x):
    #x = keras.layers.Dense(patch_projection_dim, activation=tf.nn.gelu)(x)
    #x = keras.layers.Dense(patch_projection_dim, activation=tf.nn.gelu)(x)

    x = keras.layers.Dense(patch_projection_dim, activation="swish")(x)
    x = keras.layers.Dense(patch_projection_dim, activation="swish")(x)
```

```

x = keras.layers.Dropout(0.1)(x)

return x

```

## 24.5 ViT encoder

```

[ ]: def build_vit_encoder(*,
                         image_shape,
                         #patch_size,
                         #n_patches,
                         #projection_dim
                         ):

    inputs = keras.layers.Input(shape=image_shape)
    patches = Patches(patch_size)(inputs)
    encoded_patches = PatchEncoder(n_patches, patch_projection_dim)(patches)

    for _ in range(3):
        # Layer normalization 1.
        x1 = keras.layers.LayerNormalization(epsilon = 1e-6)(encoded_patches)

        # Create a multi-head attention layer.
        attention_output = keras.layers.MultiHeadAttention(
            num_heads = n_heads, key_dim = patch_projection_dim, dropout=0.1
        )(x1, x1)

        # Skip connection 1.
        x2 = keras.layers.Add()([attention_output, encoded_patches])

        # Layer normalization 2.
        x3 = keras.layers.LayerNormalization(epsilon=1e-6)(x2)

        # MLP.
        #x3 = vit_mlp(x3, hidden_units=transformer_units, dropout_rate=0.1)
        x3 = vit_mlp(x3)

        # Skip connection 2.
        encoded_patches = keras.layers.Add()([x3, x2])

    # Create a [batch_size, projection_dim] tensor.
    representation = keras.layers.
    ↪LayerNormalization(epsilon=1e-6)(encoded_patches)
    representation = keras.layers.Flatten()(representation)
    representation = keras.layers.Dropout(0.5)(representation)

```

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

return model
```

## 24.6 Building model

```
[341]: contrastive_vit_encoder = build_vit_encoder(image_shape)
contrastive_vit_encoder.trainable = True

contrastive_vit_imgaug = CustomisedTrainImageAugmenter(**imgaug_params)

contrastive_vit_model = ContrastiveModel(
    temperature = temperature,
    encoder = contrastive_vit_encoder,
    projhead_hidden_dims = (512, 512, ),
    projhead_hidden_activation = "swish",
    projhead_output_dim = 512,
    image_augmenter = contrastive_vit_imgaug,
    n_classes = n_classes
)

contrastive_vit_model.compile(
    contrastive_optimizer = keras.optimizers.Adam(learning_rate = 0.
→001),
    probe_optimizer = keras.optimizers.Adam(),
)
```

## 24.7 Training

```
[342]: train_evaluate_and_save_models(
    model = contrastive_vit_model,
    encoder = contrastive_vit_encoder,
    train_dataset = train_dataset,
    test_dataset = test_dataset,
    epochs = n_epochs,
    savename = "contrastive_vit",
)
```

Training ...

```
Epoch 1/20
241/241 [=====] - 425s 2s/step - contrastive_loss:
```

```
2.2592 - contrastive_accuracy: 0.6784 - contrastive_accuracy_top-5: 0.8165 -
probe_loss: 6.0214 - probe_accuracy: 0.0683 - probe_accuracy_top-5: 0.2238 -
val_contrastive_accuracy_top-5: 0.9962 - val_probe_loss: 4.3649 -
val_probe_accuracy: 0.1217 - val_probe_accuracy_top-5: 0.3758
Epoch 2/20
241/241 [=====] - 447s 2s/step - contrastive_loss:
1.2982 - contrastive_accuracy: 0.9596 - contrastive_accuracy_top-5: 0.9981 -
probe_loss: 3.2758 - probe_accuracy: 0.2435 - probe_accuracy_top-5: 0.6194 -
val_contrastive_accuracy_top-5: 0.9975 - val_probe_loss: 4.4618 -
val_probe_accuracy: 0.1648 - val_probe_accuracy_top-5: 0.4796
Epoch 3/20
241/241 [=====] - 459s 2s/step - contrastive_loss:
1.2178 - contrastive_accuracy: 0.9711 - contrastive_accuracy_top-5: 0.9990 -
probe_loss: 2.9967 - probe_accuracy: 0.3239 - probe_accuracy_top-5: 0.7285 -
val_contrastive_accuracy_top-5: 0.9988 - val_probe_loss: 4.8128 -
val_probe_accuracy: 0.2039 - val_probe_accuracy_top-5: 0.5483
Epoch 4/20
241/241 [=====] - 436s 2s/step - contrastive_loss:
1.1821 - contrastive_accuracy: 0.9770 - contrastive_accuracy_top-5: 0.9994 -
probe_loss: 3.1125 - probe_accuracy: 0.3584 - probe_accuracy_top-5: 0.7618 -
val_contrastive_accuracy_top-5: 0.9990 - val_probe_loss: 5.1279 -
val_probe_accuracy: 0.2638 - val_probe_accuracy_top-5: 0.6221
Epoch 5/20
241/241 [=====] - 478s 2s/step - contrastive_loss:
1.1509 - contrastive_accuracy: 0.9784 - contrastive_accuracy_top-5: 0.9996 -
probe_loss: 3.2500 - probe_accuracy: 0.3891 - probe_accuracy_top-5: 0.7905 -
val_contrastive_accuracy_top-5: 0.9992 - val_probe_loss: 5.7315 -
val_probe_accuracy: 0.2294 - val_probe_accuracy_top-5: 0.5852
Epoch 6/20
241/241 [=====] - 416s 2s/step - contrastive_loss:
1.1209 - contrastive_accuracy: 0.9826 - contrastive_accuracy_top-5: 0.9998 -
probe_loss: 3.2871 - probe_accuracy: 0.4217 - probe_accuracy_top-5: 0.8121 -
val_contrastive_accuracy_top-5: 0.9992 - val_probe_loss: 6.6449 -
val_probe_accuracy: 0.2743 - val_probe_accuracy_top-5: 0.6377
Epoch 7/20
241/241 [=====] - 417s 2s/step - contrastive_loss:
1.0928 - contrastive_accuracy: 0.9865 - contrastive_accuracy_top-5: 0.9999 -
probe_loss: 3.4034 - probe_accuracy: 0.4414 - probe_accuracy_top-5: 0.8326 -
val_contrastive_accuracy_top-5: 0.9996 - val_probe_loss: 7.6489 -
val_probe_accuracy: 0.1931 - val_probe_accuracy_top-5: 0.5257
Epoch 8/20
241/241 [=====] - 428s 2s/step - contrastive_loss:
1.0716 - contrastive_accuracy: 0.9884 - contrastive_accuracy_top-5: 0.9999 -
probe_loss: 3.3560 - probe_accuracy: 0.4787 - probe_accuracy_top-5: 0.8576 -
val_contrastive_accuracy_top-5: 0.9996 - val_probe_loss: 6.6716 -
val_probe_accuracy: 0.3184 - val_probe_accuracy_top-5: 0.6782
Epoch 9/20
241/241 [=====] - 429s 2s/step - contrastive_loss:
```

```
1.0534 - contrastive_accuracy: 0.9891 - contrastive_accuracy_top-5: 0.9998 -
probe_loss: 3.5097 - probe_accuracy: 0.4984 - probe_accuracy_top-5: 0.8720 -
val_contrastive_accuracy_top-5: 0.9996 - val_probe_loss: 7.1888 -
val_probe_accuracy: 0.3377 - val_probe_accuracy_top-5: 0.6883
Epoch 10/20
241/241 [=====] - 429s 2s/step - contrastive_loss:
1.0419 - contrastive_accuracy: 0.9895 - contrastive_accuracy_top-5: 1.0000 -
probe_loss: 3.7257 - probe_accuracy: 0.5110 - probe_accuracy_top-5: 0.8792 -
val_contrastive_accuracy_top-5: 0.9995 - val_probe_loss: 7.2412 -
val_probe_accuracy: 0.3578 - val_probe_accuracy_top-5: 0.7127
Epoch 11/20
241/241 [=====] - 427s 2s/step - contrastive_loss:
1.0272 - contrastive_accuracy: 0.9900 - contrastive_accuracy_top-5: 0.9999 -
probe_loss: 3.7500 - probe_accuracy: 0.5379 - probe_accuracy_top-5: 0.8956 -
val_contrastive_accuracy_top-5: 0.9998 - val_probe_loss: 8.5597 -
val_probe_accuracy: 0.3270 - val_probe_accuracy_top-5: 0.6936
Epoch 12/20
241/241 [=====] - 440s 2s/step - contrastive_loss:
1.0165 - contrastive_accuracy: 0.9914 - contrastive_accuracy_top-5: 1.0000 -
probe_loss: 3.9947 - probe_accuracy: 0.5454 - probe_accuracy_top-5: 0.9026 -
val_contrastive_accuracy_top-5: 0.9997 - val_probe_loss: 8.7870 -
val_probe_accuracy: 0.3682 - val_probe_accuracy_top-5: 0.7234
Epoch 13/20
241/241 [=====] - 455s 2s/step - contrastive_loss:
1.0045 - contrastive_accuracy: 0.9912 - contrastive_accuracy_top-5: 1.0000 -
probe_loss: 3.9589 - probe_accuracy: 0.5810 - probe_accuracy_top-5: 0.9191 -
val_contrastive_accuracy_top-5: 0.9997 - val_probe_loss: 8.3446 -
val_probe_accuracy: 0.4019 - val_probe_accuracy_top-5: 0.7510
Epoch 14/20
241/241 [=====] - 431s 2s/step - contrastive_loss:
0.9933 - contrastive_accuracy: 0.9916 - contrastive_accuracy_top-5: 0.9999 -
probe_loss: 3.9768 - probe_accuracy: 0.6004 - probe_accuracy_top-5: 0.9300 -
val_contrastive_accuracy_top-5: 0.9997 - val_probe_loss: 10.0748 -
val_probe_accuracy: 0.3573 - val_probe_accuracy_top-5: 0.7247
Epoch 15/20
241/241 [=====] - 463s 2s/step - contrastive_loss:
0.9830 - contrastive_accuracy: 0.9921 - contrastive_accuracy_top-5: 1.0000 -
probe_loss: 4.0690 - probe_accuracy: 0.6257 - probe_accuracy_top-5: 0.9364 -
val_contrastive_accuracy_top-5: 0.9997 - val_probe_loss: 9.9598 -
val_probe_accuracy: 0.4068 - val_probe_accuracy_top-5: 0.7429
Epoch 16/20
241/241 [=====] - 477s 2s/step - contrastive_loss:
0.9726 - contrastive_accuracy: 0.9926 - contrastive_accuracy_top-5: 1.0000 -
probe_loss: 3.8421 - probe_accuracy: 0.6582 - probe_accuracy_top-5: 0.9507 -
val_contrastive_accuracy_top-5: 0.9999 - val_probe_loss: 10.3157 -
val_probe_accuracy: 0.4177 - val_probe_accuracy_top-5: 0.7719
Epoch 17/20
241/241 [=====] - 434s 2s/step - contrastive_loss:
```

```
0.9692 - contrastive_accuracy: 0.9925 - contrastive_accuracy_top-5: 1.0000 -
probe_loss: 4.1109 - probe_accuracy: 0.6690 - probe_accuracy_top-5: 0.9537 -
val_contrastive_accuracy_top-5: 0.9995 - val_probe_loss: 11.4945 -
val_probe_accuracy: 0.4052 - val_probe_accuracy_top-5: 0.7318
Epoch 18/20
241/241 [=====] - 454s 2s/step - contrastive_loss:
0.9505 - contrastive_accuracy: 0.9934 - contrastive_accuracy_top-5: 1.0000 -
probe_loss: 3.7773 - probe_accuracy: 0.7059 - probe_accuracy_top-5: 0.9632 -
val_contrastive_accuracy_top-5: 0.9999 - val_probe_loss: 10.2274 -
val_probe_accuracy: 0.5094 - val_probe_accuracy_top-5: 0.8197
Epoch 19/20
241/241 [=====] - 450s 2s/step - contrastive_loss:
0.9465 - contrastive_accuracy: 0.9934 - contrastive_accuracy_top-5: 1.0000 -
probe_loss: 3.8877 - probe_accuracy: 0.7240 - probe_accuracy_top-5: 0.9686 -
val_contrastive_accuracy_top-5: 0.9999 - val_probe_loss: 11.3634 -
val_probe_accuracy: 0.5025 - val_probe_accuracy_top-5: 0.8178
Epoch 20/20
241/241 [=====] - 438s 2s/step - contrastive_loss:
0.9375 - contrastive_accuracy: 0.9936 - contrastive_accuracy_top-5: 1.0000 -
probe_loss: 3.9642 - probe_accuracy: 0.7361 - probe_accuracy_top-5: 0.9721 -
val_contrastive_accuracy_top-5: 0.9997 - val_probe_loss: 12.5799 -
val_probe_accuracy: 0.4657 - val_probe_accuracy_top-5: 0.7788
```

Saving models ...

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.

WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.

WARNING:absl:Found untraced functions such as dense\_63\_layer\_call\_fn, dense\_63\_layer\_call\_and\_return\_conditional\_losses, embedding\_1\_layer\_call\_fn, embedding\_1\_layer\_call\_and\_return\_conditional\_losses, query\_layer\_call\_fn while saving (showing 5 of 100). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to:  
models/contrastive\_vit/encoder.saved\_model/assets  
  
/usr/local/anaconda3/lib/python3.8/site-packages/tensorflow/python/keras/utils/generic\_utils.py:494: CustomMaskWarning: Custom mask layers require a config and must override get\_config. When loading, the custom mask layer must be passed to the custom\_objects argument.  
    warnings.warn('Custom mask layers require a config and must override '  
INFO:tensorflow:Assets written to:  
models/contrastive\_vit/encoder.saved\_model/assets  
WARNING:absl:Found untraced functions such as train\_imgaug\_layer\_layer\_call\_fn,

```

train_imgaug_layer_layer_call_and_return_conditional_losses,
train_imgaug_layer_layer_call_fn,
train_imgaug_layer_layer_call_and_return_conditional_losses,
train_imgaug_layer_layer_call_and_return_conditional_losses while saving
(showing 5 of 105). These functions will not be directly callable after loading.
/usr/local/anaconda3/lib/python3.8/site-
packages/tensorflow/python/keras/utils/generic_utils.py:494: CustomMaskWarning:
Custom mask layers require a config and must override get_config. When loading,
the custom mask layer must be passed to the custom_objects argument.
    warnings.warn('Custom mask layers require a config and must override '
INFO:tensorflow:Assets written to:
models/contrastive_vit/main.saved_model/assets

INFO:tensorflow:Assets written to:
models/contrastive_vit/main.saved_model/assets

```

Saving training history data ...

Results:

```

Maximal contrastive_accuracy: 0.99360
Maximal contrastive_accuracy_top-5: 1.00000
Maximal probe_accuracy: 0.73614
Maximal probe_accuracy_top-5: 0.97208
Maximal val_contrastive_accuracy_top-5: 0.99994
Maximal val_probe_accuracy: 0.50935
Maximal val_probe_accuracy_top-5: 0.81974

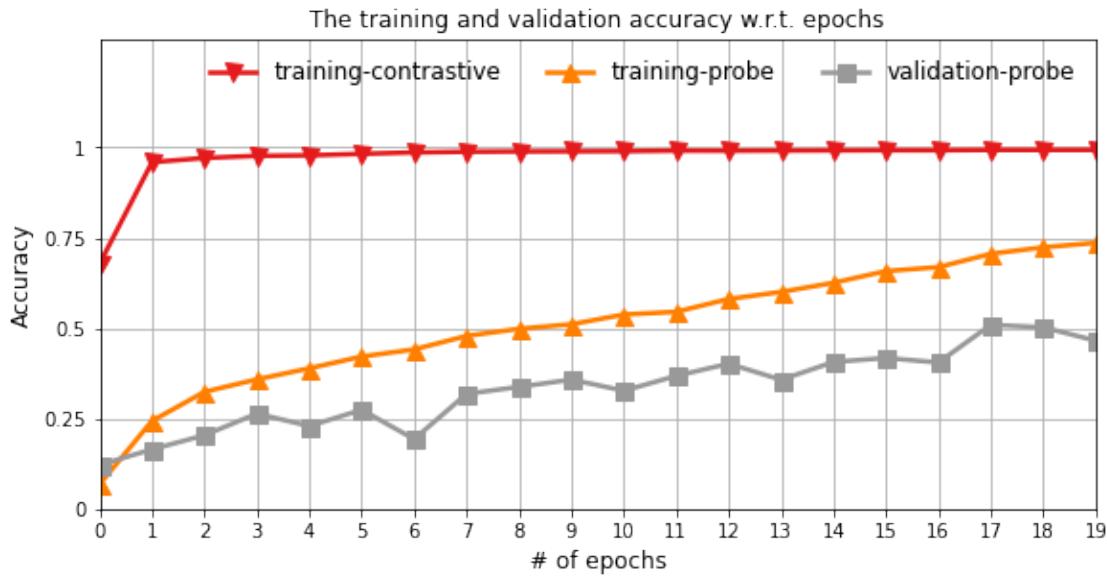
```

-- Completed --

## 24.8 Plotting results

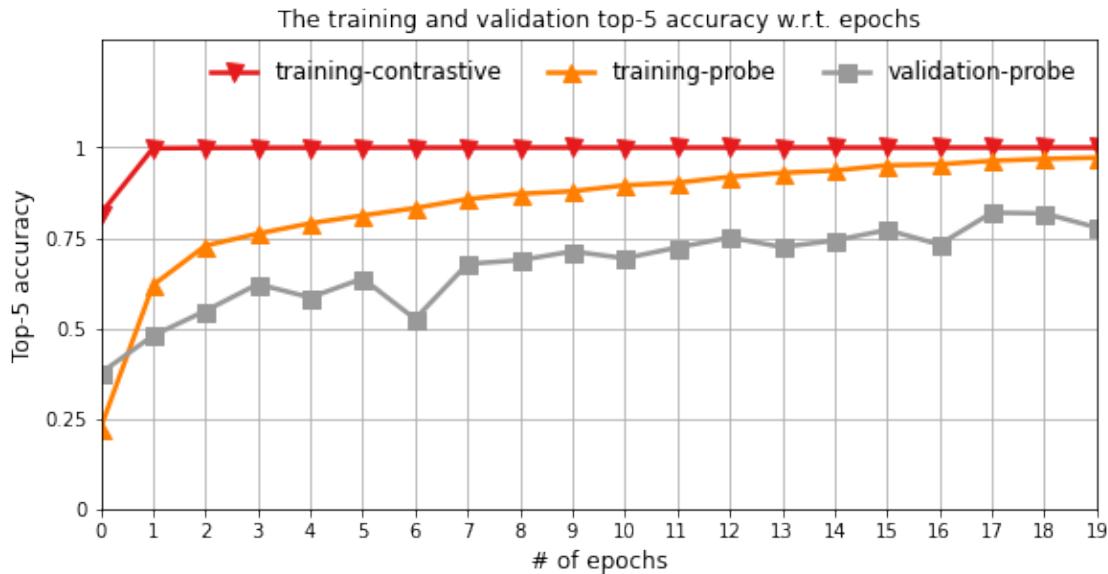
```
[28]: plot_train_results(
        resultfile = "results/contrastive_vit/history_train.pickle",
        title = "The training and validation accuracy w.r.t. epochs",
        ylabel = "Accuracy",
        keys = ["contrastive_accuracy",
                "probe_accuracy", "val_probe_accuracy"],
        labels = ["training-contrastive",
                  "training-probe", "validation-probe"],
        savename = "fig_contrastive_vit_accuracy.pdf",
        legend_location = "best",
        normalise = False
)
```

saved to: figures/fig\_contrastive\_vit\_accuracy.pdf



```
[29]: plot_train_results(
    resultfile = "results/contrastive_vit/history_train.pickle",
    title = "The training and validation top-5 accuracy w.r.t.epochs",
    ylabel = "Top-5 accuracy",
    keys = ["contrastive_accuracy_top-5",
            "probe_accuracy_top-5", "val_probe_accuracy_top-5"],
    labels = ["training-contrastive",
              "training-probe", "validation-probe"],
    savename = "fig_contrastive_vit_accuracy_top5.pdf",
    legend_location = "best",
    normalise = False
)
```

saved to: figures/fig\_contrastive\_vit\_accuracy\_top5.pdf



## 25 The facilities of plotting comparisons

```
[40]: import pickle

import matplotlib.pyplot as plt
from matplotlib import cm

from itertools import cycle

def plot_comparing_results(
    resultfiles, #a list of files
    keys,
    labels,
    title,
    ylabel,
    legend_location = "best",
    savename = None,
    normalise_data = False
):
    fig_width = 12.
```

```

fig_height = fig_width*0.5

fig = plt.figure(figsize=(fig_width, fig_height))

plt.subplots_adjust(left = 0.05,
                    right = 0.95,
                    bottom = 0.1,
                    top = 0.95)

n = len(keys)

colours = iter(cm.tab10(np.linspace(0, 1, n)))

linestyles = cycle(["-", "--", "-.", ":"])

markers = cycle(["v", "^", "s", "<", ">", "D"])

for f, k, label in zip(resultfiles, keys, labels):

    print(f"loading {f} with key: {k} and label: {label} ...")

    if not os.path.exists(f):
        continue

    history = pickle.load(open(f, "rb"))

    if k not in history:
        print(f"key {k} not found")
        continue

    v = history[k]

    if normalise_data:
        v = np.array(v) / max(v)

    c = next(colours)
    ls = next(linestyles)
    m = next(markers)

    plt.plot(
        v,
        color = c,
        label = label,
        linewidth = 2.5,

```

```

        linestyle = ls,
        marker = m,
        markersize = 9
    )

plt.xlabel("# of epochs", fontsize = 12)

plt.ylabel(ylabel, fontsize = 12)

#k = keys[0]
#v = history[k]
plt.xlim(0, len(v) - 1)
plt.ylim(0, 1.2)

xticks = np.linspace(0,
                     len(v) - 1,
                     num = len(v),
                     endpoint = True,
                     dtype = int)
xlabels = xticks

plt.xticks(xticks, xlabels)
plt.yticks([0, 0.25, 0.5, 0.75, 1], [0, 0.25, 0.5, 0.75, 1])

plt.grid()

plt.legend(
    loc = legend_location, #'upper center',
    fontsize = 12,
    ncol = 4,
    frameon = False)

if not os.path.exists("figures"):
    os.makedirs("figures")

if savename:
    figpath = "figures" + os.path.sep + savename
    plt.savefig(figpath,
                dpi=600,
                format='pdf')

    print("saved to: ", figpath)

plt.title(title)

plt.show()

```

## 26 Comparing training probe top-1 accuracy

```
[483]: keys = [
    "accuracy",
    "accuracy",
    "accuracy",
    "accuracy",

    "probe_accuracy",
    "probe_accuracy",
    "probe_accuracy",
]

resultfiles = [
    "results/baseline_simple/history_train.pickle",
    "results/baseline_vgg16/history_train.pickle",
    "results/baseline_resnet/history_train.pickle",
    "results/baseline_mobilenet/history_train.pickle",

    "results/contrastive_simple/history_train.pickle",
    "results/contrastive_vgg16/history_train.pickle",
    "results/contrastive_vit/history_train.pickle",
]

labels = [
    "baseline-simple",
    "baseline-vgg16",
    "baseline-resnet50",
    "baseline-mobilenet",

    "contrastive-simple",
    "contrastive-vgg16",
    "contrastive-vit",
]

plot_comparing_results(
    resultfiles = resultfiles,
    keys = keys,
    labels = labels,

    title = "The training accuracy comparision w.r.
→t. epochs",
    ylabel = "Accuracy",

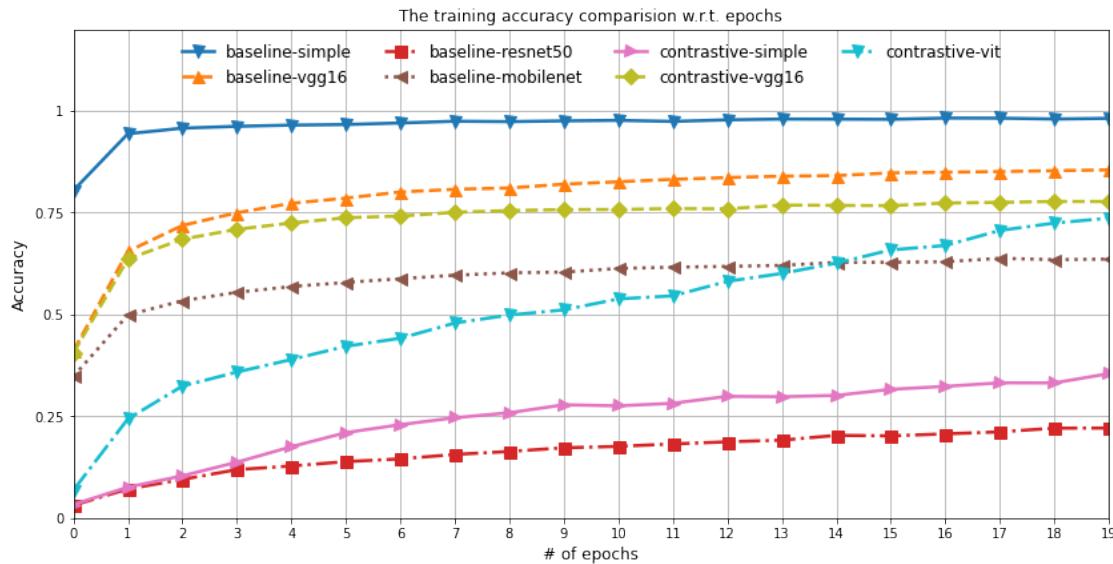
    legend_location = "upper center",
```

```

    savename = "fig_training_accuracy_comparison.
    ↪pdf",
    normalise_data = False
)

```

loading results/baseline\_simple/history\_train.pickle with key: accuracy and  
 label: baseline-simple ...  
 loading results/baseline\_vgg16/history\_train.pickle with key: accuracy and  
 label: baseline-vgg16 ...  
 loading results/baseline\_resnet/history\_train.pickle with key: accuracy and  
 label: baseline-resnet50 ...  
 loading results/baseline\_mobilenet/history\_train.pickle with key: accuracy and  
 label: baseline-mobilenet ...  
 loading results/contrastive\_simple/history\_train.pickle with key: probe\_accuracy  
 and label: contrastive-simple ...  
 loading results/contrastive\_vgg16/history\_train.pickle with key: probe\_accuracy  
 and label: contrastive-vgg16 ...  
 loading results/contrastive\_vit/history\_train.pickle with key: probe\_accuracy  
 and label: contrastive-vit ...  
 saved to: figures/fig\_training\_accuracy\_comparison.pdf



## 27 Comparing validation probe top-1 accuracy

```

[484]: keys = [
    "val_accuracy",
    "val_accuracy",
]

```

```

    "val_accuracy",
    "val_accuracy",

    "val_probe_accuracy",
    "val_probe_accuracy",
    "val_probe_accuracy",
]

resultfiles = [
    "results/baseline_simple/history_train.pickle",
    "results/baseline_vgg16/history_train.pickle",
    "results/baseline_resnet/history_train.pickle",
    "results/baseline_mobilenet/history_train.pickle",

    "results/contrastive_simple/history_train.pickle",
    "results/contrastive_vgg16/history_train.pickle",
    "results/contrastive_vit/history_train.pickle",
]

labels = [
    "baseline-simple",
    "baseline-vgg16",
    "baseline-resnet50",
    "baseline-mobilenet",

    "contrastive-simple",
    "contrastive-vgg16",
    "contrastive-vit",
]

plot_comparing_results(
    resultfiles = resultfiles,
    keys = keys,
    labels = labels,

    title = "The validation accuracy comparision w.
→r.t. epochs",
    ylabel = "Accuracy",

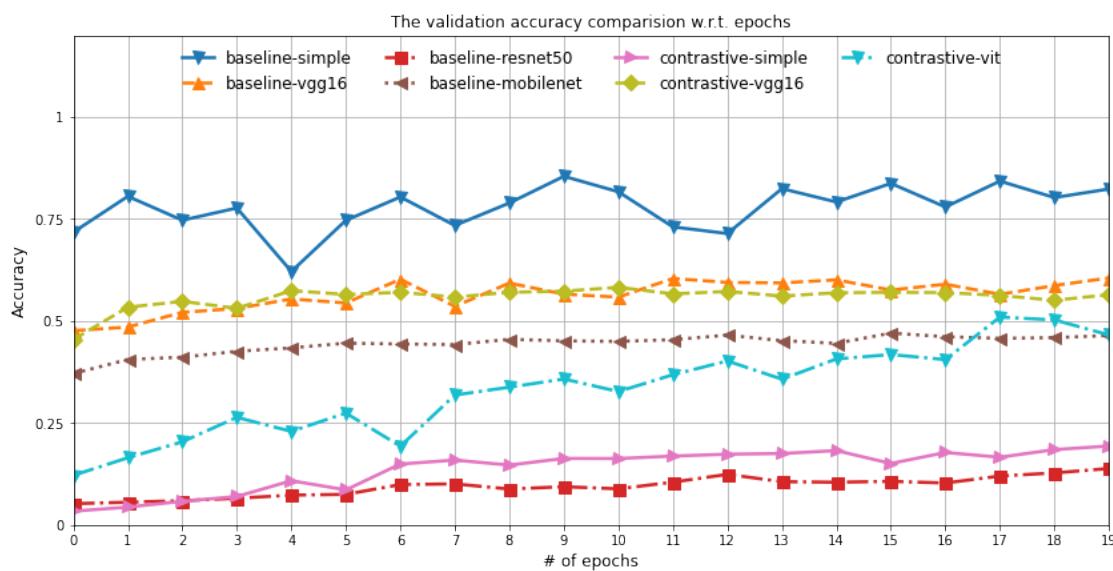
    legend_location = "upper center",
    savename = "fig_validation_accuracy_comparison.
→pdf",
    normalise_data = False
)

```

```

loading results/baseline_simple/history_train.pickle with key: val_accuracy and
label: baseline-simple ...
loading results/baseline_vgg16/history_train.pickle with key: val_accuracy and
label: baseline-vgg16 ...
loading results/baseline_resnet/history_train.pickle with key: val_accuracy and
label: baseline-resnet50 ...
loading results/baseline_mobilenet/history_train.pickle with key: val_accuracy and
label: baseline-mobilenet ...
loading results/contrastive_simple/history_train.pickle with key:
val_probe_accuracy and label: contrastive-simple ...
loading results/contrastive_vgg16/history_train.pickle with key:
val_probe_accuracy and label: contrastive-vgg16 ...
loading results/contrastive_vit/history_train.pickle with key:
val_probe_accuracy and label: contrastive-vit ...
saved to: figures/fig_validation_accuracy_comparison.pdf

```



## 28 Comparing training probe top-5 accuracy

```
[485]: keys = [
    "accuracy_top-5",
    "accuracy_top-5",
    "accuracy_top-5",
    "accuracy_top-5",
    "probe_accuracy_top-5",
    "probe_accuracy_top-5",
    "probe_accuracy_top-5",
```

```

]

resultfiles = [
    "results/baseline_simple/history_train.pickle",
    "results/baseline_vgg16/history_train.pickle",
    "results/baseline_resnet/history_train.pickle",
    "results/baseline_mobilenet/history_train.pickle",

    "results/contrastive_simple/history_train.pickle",
    "results/contrastive_vgg16/history_train.pickle",
    "results/contrastive_vit/history_train.pickle",
]

labels = [
    "baseline-simple",
    "baseline-vgg16",
    "baseline-resnet50",
    "baseline-mobilenet",

    "contrastive-simple",
    "contrastive-vgg16",
    "contrastive-vit",
]

plot_comparing_results(
    resultfiles = resultfiles,
    keys = keys,
    labels = labels,

    title = "The training top-5 accuracy\u2192comparision w.r.t. epochs",
    ylabel = "Top-5 accuracy",
    legend_location = "upper center",
    savename = \u2192"fig_training_accuracy_comparison_top5.pdf",
    normalise_data = False
)

```

```

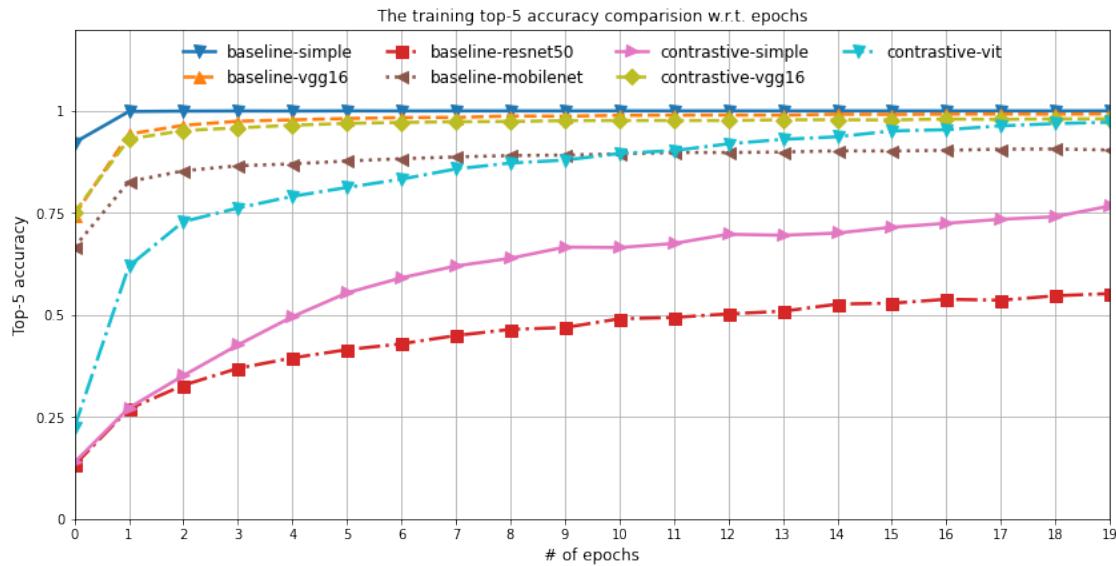
loading results/baseline_simple/history_train.pickle with key: accuracy_top-5
and label: baseline-simple ...
loading results/baseline_vgg16/history_train.pickle with key: accuracy_top-5 and
label: baseline-vgg16 ...
loading results/baseline_resnet/history_train.pickle with key: accuracy_top-5
and label: baseline-resnet50 ...

```

```

loading results/baseline_mobilenet/history_train.pickle with key: accuracy_top-5
and label: baseline-mobilenet ...
loading results/contrastive_simple/history_train.pickle with key:
probe_accuracy_top-5 and label: contrastive-simple ...
loading results/contrastive_vgg16/history_train.pickle with key:
probe_accuracy_top-5 and label: contrastive-vgg16 ...
loading results/contrastive_vit/history_train.pickle with key:
probe_accuracy_top-5 and label: contrastive-vit ...
saved to: figures/fig_training_accuracy_comparison_top5.pdf

```



## 29 Comparing validation probe top-5 accuracy

```

[486]: keys = [
    "val_accuracy_top-5",
    "val_accuracy_top-5",
    "val_accuracy_top-5",
    "val_accuracy_top-5",
    "val_probe_accuracy_top-5",
    "val_probe_accuracy_top-5",
    "val_probe_accuracy_top-5",
]

resultfiles = [
    "results/baseline_simple/history_train.pickle",
    "results/baseline_vgg16/history_train.pickle",
    "results/baseline_resnet/history_train.pickle",
]

```

```

        "results/baseline_mobilenet/history_train.pickle",
        "results/contrastive_simple/history_train.pickle",
        "results/contrastive_vgg16/history_train.pickle",
        "results/contrastive_vit/history_train.pickle",
    ]

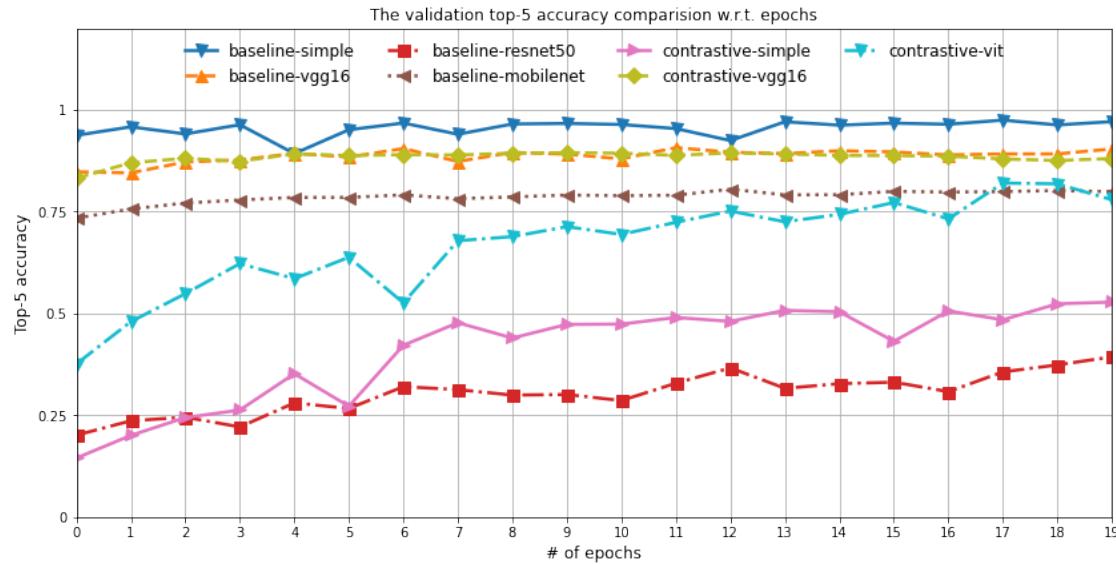
labels = [
    "baseline-simple",
    "baseline-vgg16",
    "baseline-resnet50",
    "baseline-mobilenet",
    "contrastive-simple",
    "contrastive-vgg16",
    "contrastive-vit",
]
]

plot_comparing_results(
    resultfiles = resultfiles,
    keys = keys,
    labels = labels,
    title = "The validation top-5 accuracy\u2192comparision w.r.t. epochs",
    ylabel = "Top-5 accuracy",
    legend_location = "upper center",
    savename = \u2192"fig_validation_accuracy_comparison_top5.pdf",
    normalise_data = False
)

```

loading results/baseline\_simple/history\_train.pickle with key:  
val\_accuracy\_top-5 and label: baseline-simple ...  
loading results/baseline\_vgg16/history\_train.pickle with key: val\_accuracy\_top-5  
and label: baseline-vgg16 ...  
loading results/baseline\_resnet/history\_train.pickle with key:  
val\_accuracy\_top-5 and label: baseline-resnet50 ...  
loading results/baseline\_mobilenet/history\_train.pickle with key:  
val\_accuracy\_top-5 and label: baseline-mobilenet ...  
loading results/contrastive\_simple/history\_train.pickle with key:  
val\_probe\_accuracy\_top-5 and label: contrastive-simple ...  
loading results/contrastive\_vgg16/history\_train.pickle with key:  
val\_probe\_accuracy\_top-5 and label: contrastive-vgg16 ...

```
loading results/contrastive_vit/history_train.pickle with key:
val_probe_accuracy_top-5 and label: contrastive-vit ...
saved to: figures/fig_validation_accuracy_comparison_top5.pdf
```



## 30 Comparing training probe top-1 accuracy w/ and w/o temperature scheduling

```
[41]: keys = [
    "probe_accuracy",
    "probe_accuracy",
]

resultfiles = [
    "results/contrastive_simple/history_train.pickle",
    "results/contrastive_simple_dynamic/history_train.pickle",
]

labels = [
    "contrastive-simple-static",
    "contrastive-simple-dynamic",
]

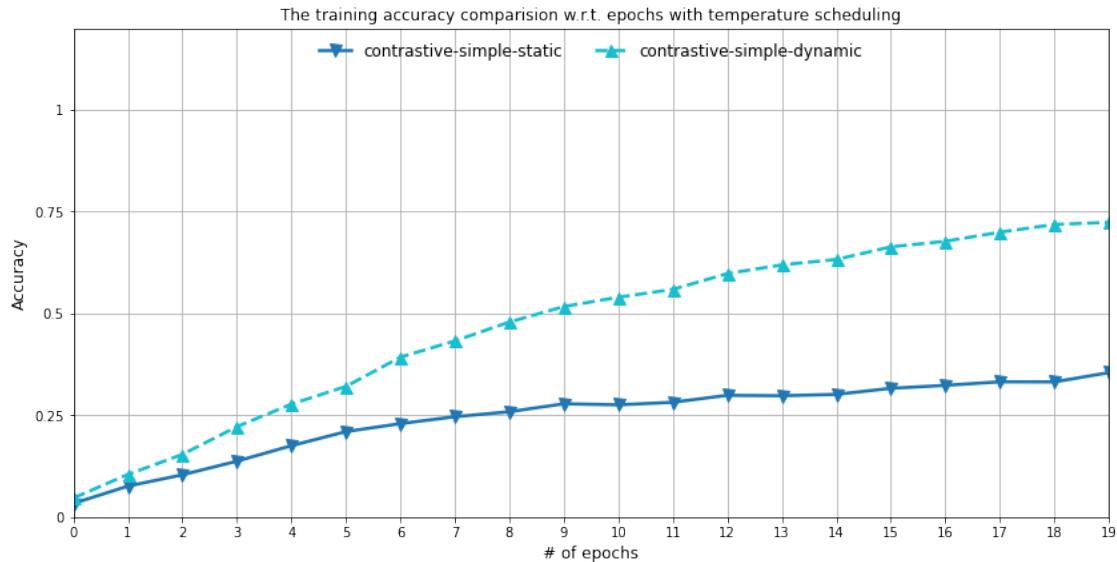
plot_comparing_results(
    resultfiles = resultfiles,
    keys = keys,
    labels = labels,
```

```

        title = "The training accuracy comparision w.r.
        ↵t. epochs with temperature scheduling",
        ylabel = "Accuracy",
        legend_location = "upper center",
        savename = █
        ↵"fig_training_accuracy_comparison_temperature.pdf",
        normalise_data = False
    )

```

loading results/contrastive\_simple/history\_train.pickle with key: probe\_accuracy  
and label: contrastive-simple-static ...  
loading results/contrastive\_simple\_dynamic/history\_train.pickle with key:  
probe\_accuracy and label: contrastive-simple-dynamic ...  
saved to: figures/fig\_training\_accuracy\_comparison\_temperature.pdf



### 31 Comparing validation probe top-1 accuracy w/ and w/o temperature scheduling

```
[42]: keys = [
    "val_probe_accuracy",
    "val_probe_accuracy",
]
resultfiles = [
```

```

        "results/contrastive_simple/history_train.pickle",
        "results/contrastive_simple_dynamic/history_train.pickle",
    ]

labels = [
    "contrastive-simple-static",
    "contrastive-simple-dynamic",
]

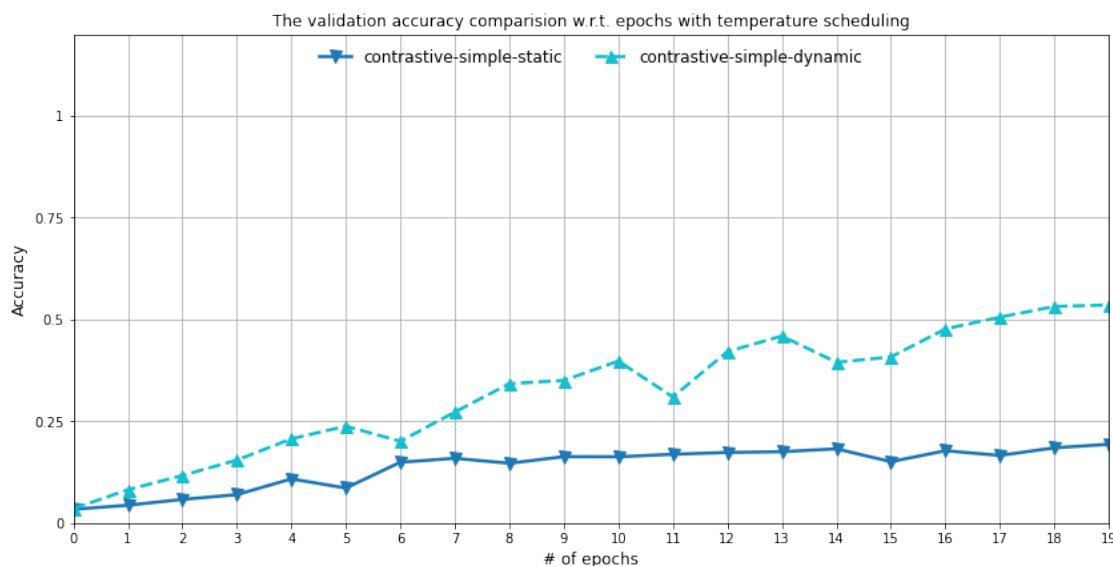
plot_comparing_results(
    resultfiles = resultfiles,
    keys = keys,
    labels = labels,

    title = "The validation accuracy comparision w.r.t. epochs with temperature scheduling",
    ylabel = "Accuracy",

    legend_location = "upper center",
    savename = "fig_validation_accuracy_comparison_temperature.pdf",
    normalise_data = False
)

```

loading results/contrastive\_simple/history\_train.pickle with key:  
val\_probe\_accuracy and label: contrastive-simple-static ...  
loading results/contrastive\_simple\_dynamic/history\_train.pickle with key:  
val\_probe\_accuracy and label: contrastive-simple-dynamic ...  
saved to: figures/fig\_validation\_accuracy\_comparison\_temperature.pdf



## 32 Comparing training probe top-5 accuracy w/ and w/o temperature scheduling

```
[44]: keys = [
    "probe_accuracy_top-5",
    "probe_accuracy_top-5",
]

resultfiles = [
    "results/contrastive_simple/history_train.pickle",
    "results/contrastive_simple_dynamic/history_train.pickle",
]

labels = [
    "contrastive-simple-static",
    "contrastive-simple-dynamic",
]

plot_comparing_results(
    resultfiles = resultfiles,
    keys = keys,
    labels = labels,

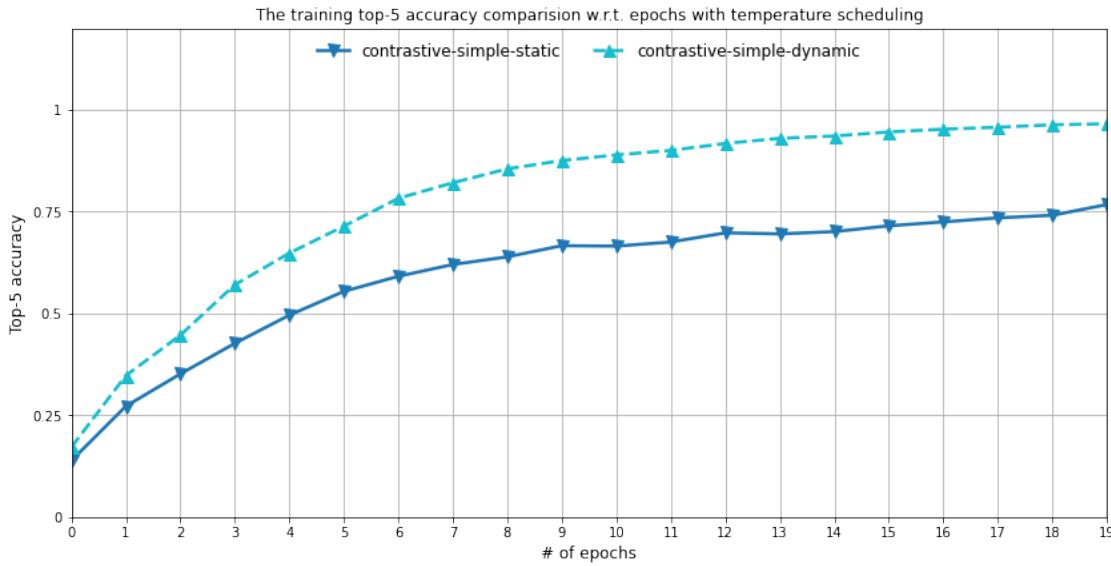
    title = "The training top-5 accuracy\u2193
    \u2193comparision w.r.t. epochs with temperature scheduling",
    ylabel = "Top-5 accuracy",

    legend_location = "upper center",

    savename = \u2193
    \u2193"fig_training_accuracy_comparison_temperature_top5.pdf",

    normalise_data = False
)
```

```
loading results/contrastive_simple/history_train.pickle with key:
probe_accuracy_top-5 and label: contrastive-simple-static ...
loading results/contrastive_simple_dynamic/history_train.pickle with key:
probe_accuracy_top-5 and label: contrastive-simple-dynamic ...
saved to: figures/fig_training_accuracy_comparison_temperature_top5.pdf
```



```
[45]: keys = [
    "val_probe_accuracy_top-5",
    "val_probe_accuracy_top-5",
]

resultfiles = [
    "results/contrastive_simple/history_train.pickle",
    "results/contrastive_simple_dynamic/history_train.pickle",
]

labels = [
    "contrastive-simple-static",
    "contrastive-simple-dynamic",
]

plot_comparing_results(
    resultfiles = resultfiles,
    keys = keys,
    labels = labels,

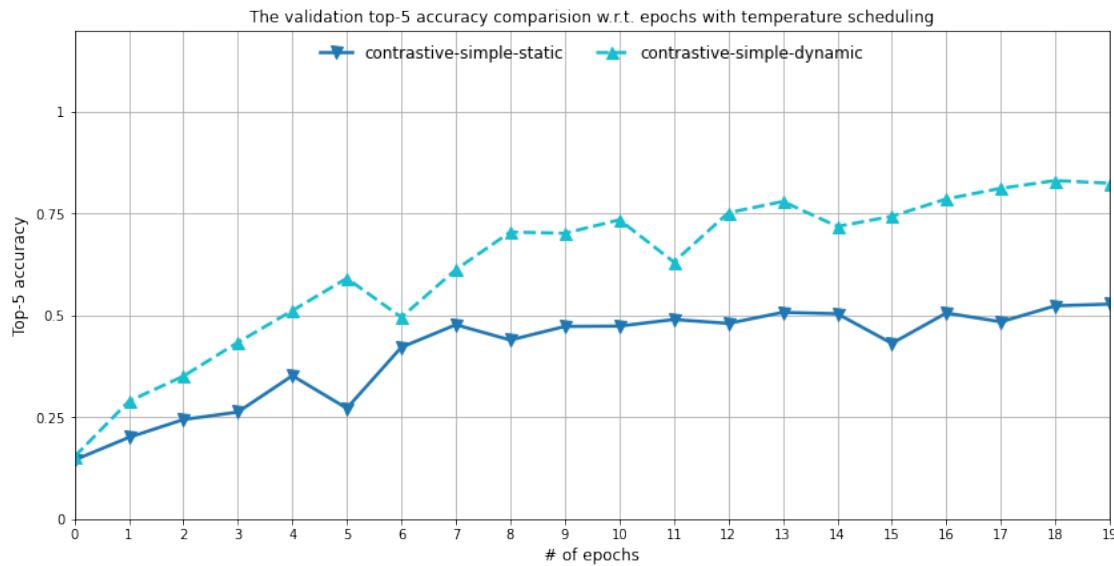
        title = "The validation top-5 accuracy\u2192comparision w.r.t. epochs with temperature scheduling",
        ylabel = "Top-5 accuracy",

        legend_location = "upper center",

        savename = \u2192"fig_validation_accuracy_comparison_temperature_top5.pdf",
```

```
    normalise_data = False  
)
```

```
loading results/contrastive_simple/history_train.pickle with key:  
val_probe_accuracy_top-5 and label: contrastive-simple-static ...  
loading results/contrastive_simple_dynamic/history_train.pickle with key:  
val_probe_accuracy_top-5 and label: contrastive-simple-dynamic ...  
saved to: figures/fig_validation_accuracy_comparison_temperature_top5.pdf
```



## 33 The facilities of plotting 2D projections

### 33.1 Plotting one model

```
[25]: import numpy as np  
import pandas as pd  
  
import matplotlib.pyplot as plt  
from matplotlib import cm  
  
import os  
import sys  
  
import random  
  
from sklearn.manifold import TSNE  
#from sklearn.metrics import pairwise_distances  
import umap
```

```

from sklearn.decomposition import PCA
from sklearn.decomposition import KernelPCA

def plot_projection_one(
    features,
    labels,
    title,
    legend_location = "best",
    savename = None,
    #kernelpca-cosine, tsne-cosine, umap-cosine
    algo = "kernelpca-cosine",
):
    fig_width = 7
    fig_height = 7

    fig = plt.figure(figsize=(fig_width, fig_height))

    plt.subplots_adjust(left = 0.05,
                        right = 0.95,
                        bottom = 0.05,
                        top = 0.95)

    #building colours
    colours_dict = {}
    colours_ = iter(cm.rainbow(np.linspace(0, 1, len(set(labels)))))

    for label in set(labels):
        colours_dict[label] = next(colours_)

    #n_classes = len(set(labels))

    #class_labels = list(set(labels))

    reducers_dict = {
        "kernelpca-cosine": KernelPCA(n_components=2,
                                       kernel = "cosine"),
    }

```

```

"tsne-cosine": TSNE( metric = "cosine",
                      square_distances = True,
                      n_iter = 15000,
                      n_jobs = -1,
                      random_state = 1133),

"umap-cosine": umap.UMAP(metric = "cosine")
}

reducer = reducers_dict[algo]

projections = reducer.fit_transform(features)

for idx, (xi, label) in enumerate(zip(projections, labels)):
    plt.scatter(xi[0],
                xi[1],
                s = 70,
                alpha = 0.5,
                edgecolors = None,
                linewidths = 0,
                color = colours_dict[label],
                label = f"class-{label}")

if not os.path.exists("figures"):
    os.makedirs("figures")

if savename:
    figpath = "figures" + os.path.sep + savename

    plt.savefig(figpath,
                dpi = 600,
                format = 'pdf')

    print("saved to: ", figpath)

plt.show()

plt.close()

```

### 33.2 Plotting model comparison

```
[37]: import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib import cm

import os
import sys

import random

from sklearn.manifold import TSNE
#from sklearn.metrics import pairwise_distances
import umap

from sklearn.decomposition import PCA
from sklearn.decomposition import KernelPCA

import math

def plot_projections(
    features_dict,
    labels,
    keys,
    title,
    subtitles,
    ncols = 2,
    savename = None,
    #kernelpca-cosine, tsne-cosine, umap-cosine
    algo = "kernelpca-cosine",
    subfig_height = 4,
    subfig_width = 4,
):
    nrows = math.ceil(len(keys) / ncols)
```

```

print(f"nrows = {nrows} ncols = {ncols}")

fig_width = ncols * subfig_width
fig_height = nrows * subfig_height

#fig = plt.figure(figsize=(fig_width, fig_height))
figsize=(fig_width, fig_height)
fig, axes = plt.subplots(nrows = nrows,
                        ncols = ncols,
                        figsize = figsize)

axes_flat = axes.flat

plt.subplots_adjust(left = 0.05,
                    right = 0.95,
                    bottom = 0.05,
                    top = 0.95)

#building colours
colours_dict = {}
colours_ = iter(cm.rainbow(np.linspace(0, 1, len(set(labels)))))

for label in set(labels):
    colours_dict[label] = next(colours_)

#n_classes = len(set(labels_))
#class_labels = list(set(labels))

reducers_dict = {
    "kernelpca-cosine": KernelPCA(n_components=2,
                                    kernel = "cosine"),

    "tsne-cosine": TSNE( metric = "cosine",
                         square_distances = True,
                         n_iter = 15000,
                         n_jobs = -1,
                         random_state = 1133),

    "umap-cosine": umap.UMAP(metric = "cosine")
}

reducer = reducers_dict[algo]

```

```

projections_dict = {}

for key in keys:
    features = features_dict[key]
    projections = reducer.fit_transform(features)
    projections_dict[key] = projections


for row in range(nrows):
    for col in range(ncols):

        key_idx = row * ncols + col

        if key_idx >= len(keys):
            continue

        key = keys[key_idx]

        projections = projections_dict[key]

        ax = axes_flat[key_idx]

        subtitle = subtitles[key_idx]

        ax.title.set_text(subtitle)

        for idx, (xi, label) in enumerate(zip(projections, labels)):
            label = labels[idx]
            ax.scatter(
                xi[0],
                xi[1],
                s = 70,
                alpha = 0.5,
                edgecolors = None,
                linewidths = 0,
                color = colours_dict[label],
                label = f"class-{label}"
            )

if not os.path.exists("figures"):
    os.makedirs("figures")

if savename:
    figpath = "figures" + os.path.sep + savename

plt.savefig(figpath,

```

```

        dpi = 600,
        format = 'pdf')

print("saved to: ", figpath)

plt.show()

plt.close()

```

## 34 Sampling images for projection visualisation

```
[11]: def sample_projection_images(
    dataset,
    n_classes = 6,
    n_images_per_class = 20,
):
    # convert tensorflow dataset to numpy
    X = []
    y = []

    for xi, yi in dataset.as_numpy_iterator():
        X = X + list(xi)
        y = y + list(yi)

    X = np.array(X, dtype = object)
    y = np.array(y, dtype = object)

    print("X.shape = ", X.shape)
    print("y.shape = ", y.shape)

    # randomly choose classes
    chosen_classes = np.random.choice(list(set(y)),
                                       size = n_classes,
                                       replace = False)
    print("chosen_classes = ", chosen_classes)

    images = []
    labels = []

    for label in chosen_classes:

```

```

indices = np.where(y == label)
indices = indices[0]

if len(indices) < n_images_per_class:
    size = len(indices)
else:
    size = n_images_per_class

#print(size)

chosen_indices = np.random.choice(list(indices),
                                   size = size,
                                   replace = False)

images = images + list(X[chosen_indices])
labels = labels + list(y[chosen_indices])

images = np.array(images, dtype = object)
labels = np.array(labels, dtype = object)

print("images.shape = ", images.shape)
print("labels.shape = ", labels.shape)

return images, labels

```

```

[71]: import pickle
import os

if not os.path.exists("data"):
    os.makedirs("data")

images, labels = sample_projection_images(
                    dataset = test_dataset,
                    n_classes = 6,
                    n_images_per_class = 40
                )

data = {}
data["images"] = images
data["labels"] = labels

pickle.dump(data, open("data" + os.path.sep + "projection_samples.pickle", "wb"))
del data, images, labels

```

X.shape = (7700, 64, 64, 3)

```
y.shape = (7700,)  
chosen_classes = [45 99 10 28 80 16]  
images.shape = (240, 64, 64, 3)  
labels.shape = (240,)
```

```
[28]: import tensorflow as tf  
import tensorflow_addons as tfa  
from tensorflow import keras  
  
import numpy as np  
import pandas as pd  
import os  
import sys  
import random  
  
import matplotlib.pyplot as plt  
  
import pickle  
import os  
  
data = pickle.load(open("data" + os.path.sep + "projection_samples.pickle",  
                      "rb"))  
  
images = data["images"]  
labels = data["labels"]  
  
del data
```

## 35 Projection algorithm setting

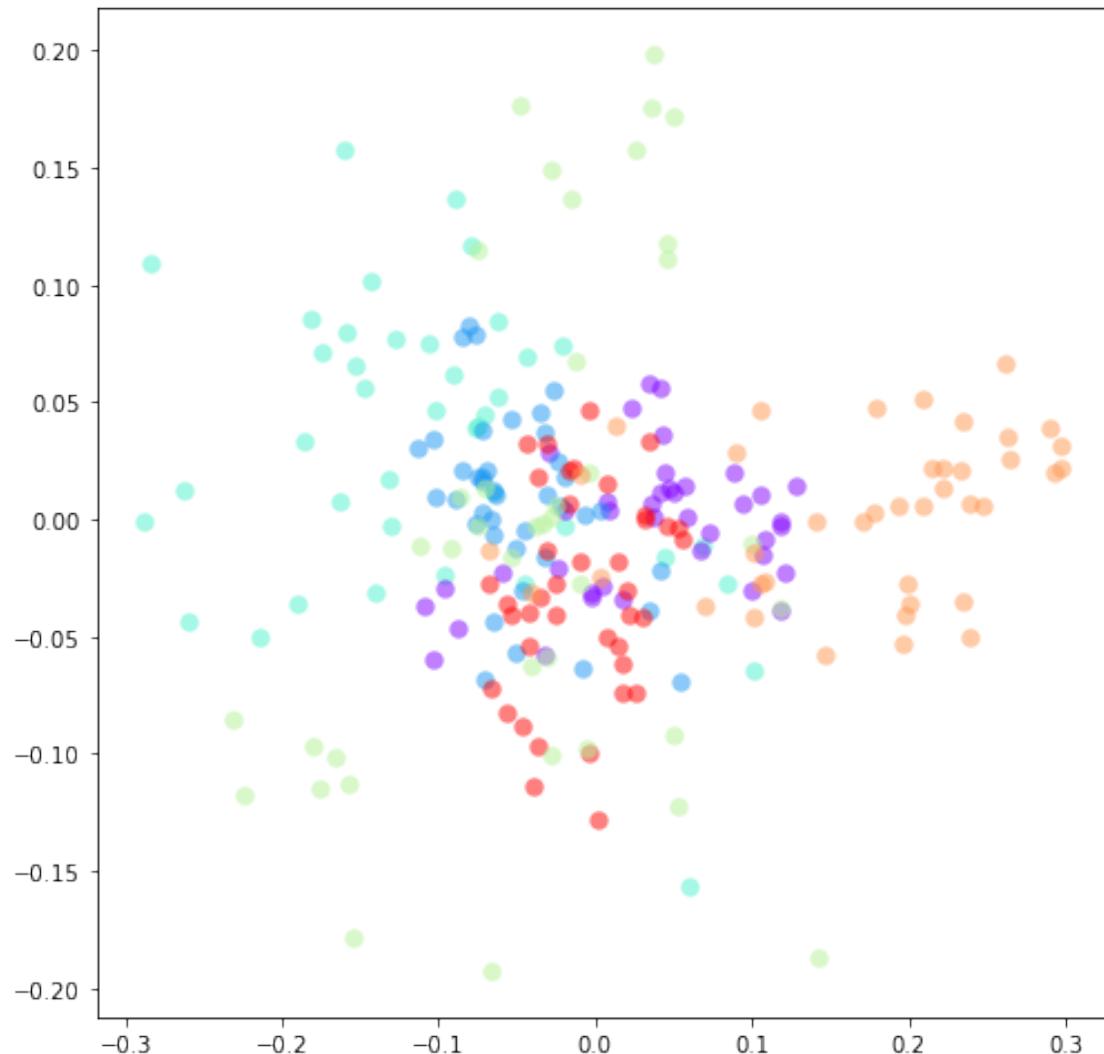
```
[26]: project_algo = "kernelpca-cosine"
```

## 36 Projecting raw images

```
[8]: n_feature_dims = np.prod(images[0].shape)  
  
features = images.reshape(-1, n_feature_dims)  
print("features.shape = ", features.shape)  
  
plot_projection_one(  
    features = features,  
    labels = labels,  
  
    title = "The 2D projection of raw images",  
  
    legend_location = "best",
```

```
savename = "fig_2d_proj_raw.pdf",  
algo = project_algo  
)
```

```
features.shape = (240, 12288)  
projections.shape = (240, 2)  
saved to: figures/fig_2d_proj_raw.pdf
```



## 37 Projecting with baseline simple model

```
[9]: model = keras.models.load_model("models/baseline_simple/main.saved_model")

images_ = images.astype("float32")
images_ = tf.convert_to_tensor(images_)

features_ = model(images_)
features = features_.numpy()

print("features.shape = ", features.shape)

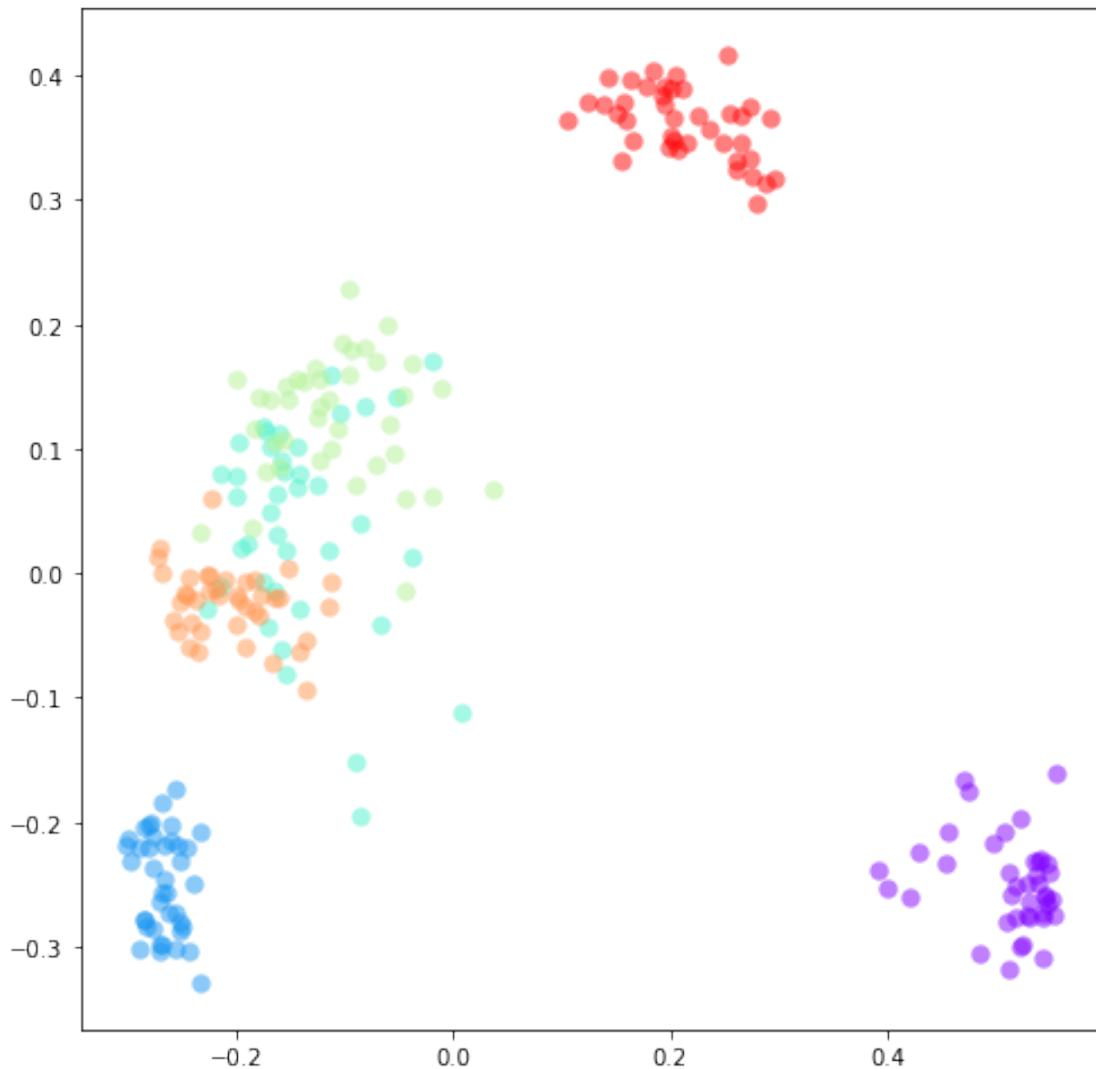
plot_projection_one(
    features = features,
    labels = labels,
    title = "The 2D projection of baseline simple model",
    legend_location = "best",
    savename = "fig_2d_proj_baseline_simple.pdf",
    algo = project_algo
)

del model
```

WARNING:absl:Importing a function  
(\_inference\_conv2d\_1\_layer\_call\_and\_return\_conditional\_losses\_28219846) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (\_inference\_baseline\_simple\_model\_layer\_call\_and\_return\_conditional\_losses\_28221690) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_inference\_encoder\_simple\_layer\_call\_and\_return\_conditional\_losses\_28222766) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_inference\_conv2d\_2\_layer\_call\_and\_return\_conditional\_losses\_28223436) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (\_inference\_\_wrapped\_model\_28218849) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (\_inference\_baseline\_simple\_model\_layer\_call\_and\_return\_conditional\_losses\_28222148) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_inference\_conv2d\_3\_layer\_call\_and\_return\_conditional\_losses\_28223461) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function

```
(__inference_conv2d_2_layer_call_and_return_conditional_losses_28219902) with
ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_conv2d_layer_call_and_return_conditional_losses_28223235) with ops
with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_conv2d_1_layer_call_and_return_conditional_losses_28223260) with
ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_conv2d_layer_call_and_return_conditional_losses_28219824) with ops
with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_encoder_simple_layer_call_and_return_conditional_losses_28222668)
with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_conv2d_3_layer_call_and_return_conditional_losses_28219924) with
ops with custom gradients. Will likely fail if a gradient is requested.

features.shape = (240, 154)
projections.shape = (240, 2)
saved to: figures/fig_2d_proj_baseline_simple.pdf
```



## 38 Projecting with baseline VGG16

```
[10]: model = keras.models.load_model("models/baseline_vgg16/main.saved_model")

images_ = images.astype("float32")
images_ = tf.convert_to_tensor(images_)

features_ = model(images_)
features = features_.numpy()

print("features.shape = ", features.shape)

plot_projection_one(
```

```

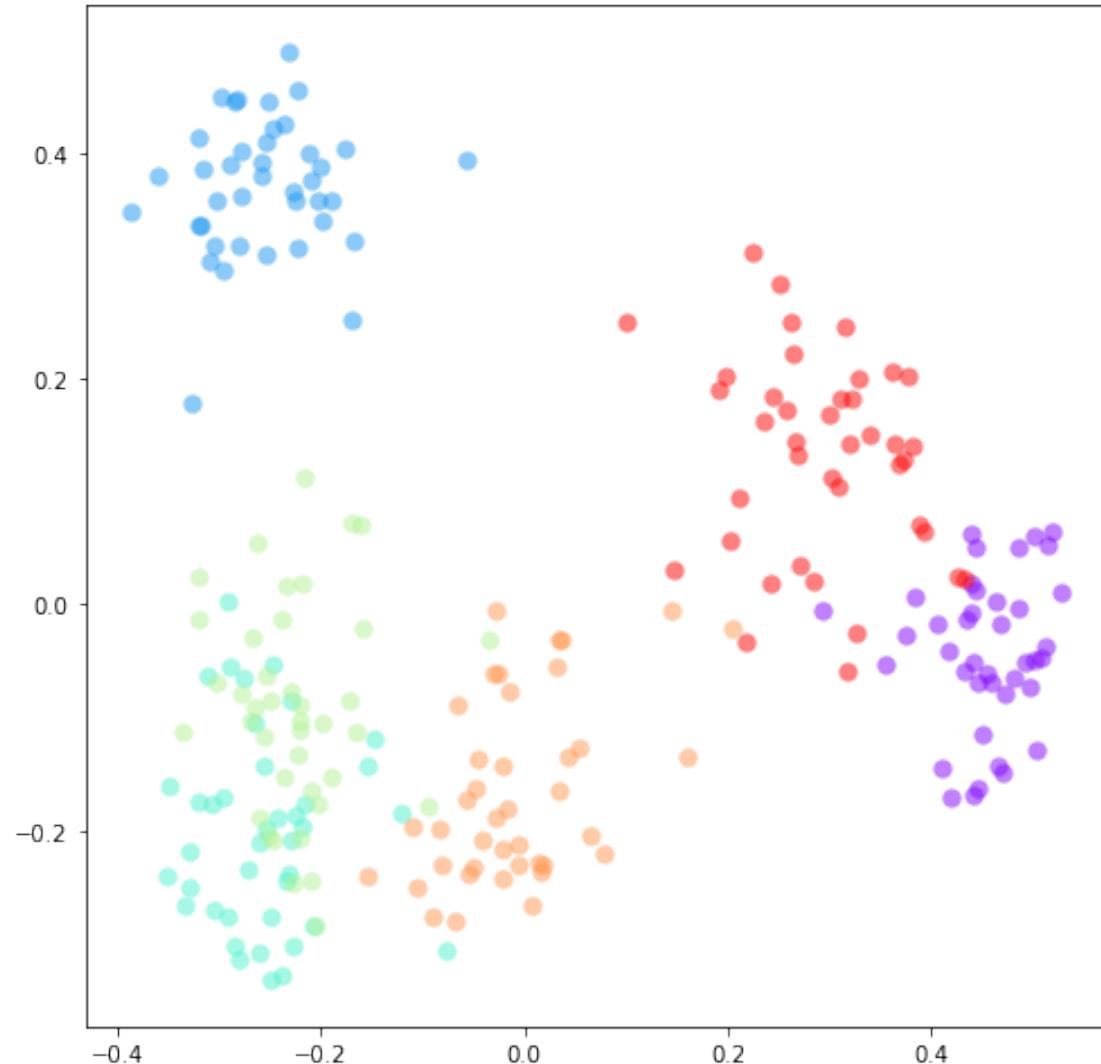
        features = features,
        labels = labels,

        title = "The 2D projection of baseline VGG16 model",
        legend_location = "best",
        savename = "fig_2d_proj_baseline_vgg16.pdf",
        algo = project_algo
    )

del model

features.shape = (240, 154)
projections.shape = (240, 2)
saved to: figures/fig_2d_proj_baseline_vgg16.pdf

```



## 39 Projecting with baseline ResNet50

```
[11]: model = keras.models.load_model("models/baseline_resnet/main.saved_model")

images_ = images.astype("float32")
images_ = tf.convert_to_tensor(images_)

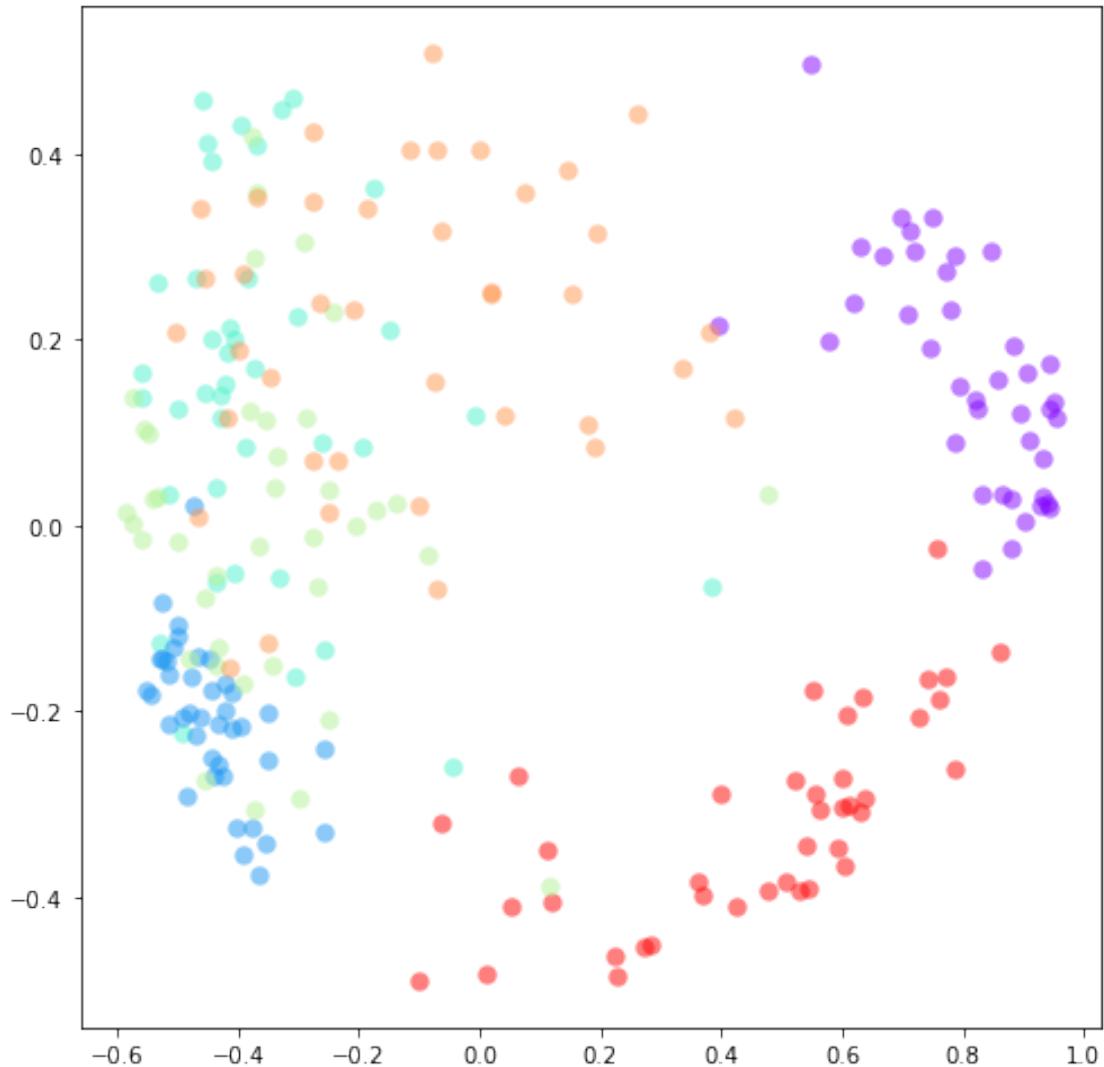
features_ = model(images_)
features = features_.numpy()

print("features.shape = ", features.shape)

plot_projection_one(
    features = features,
    labels = labels,
    title = "The 2D projection of baseline ResNet50 model",
    legend_location = "best",
    savename = "fig_2d_proj_baseline_resnet.pdf",
    algo = project_algo
)

del model

features.shape = (240, 154)
projections.shape = (240, 2)
saved to: figures/fig_2d_proj_baseline_resnet.pdf
```



## 40 Projecting with baseline MobileNet

```
[13]: model = keras.models.load_model("models/baseline_mobilenet/main.saved_model")

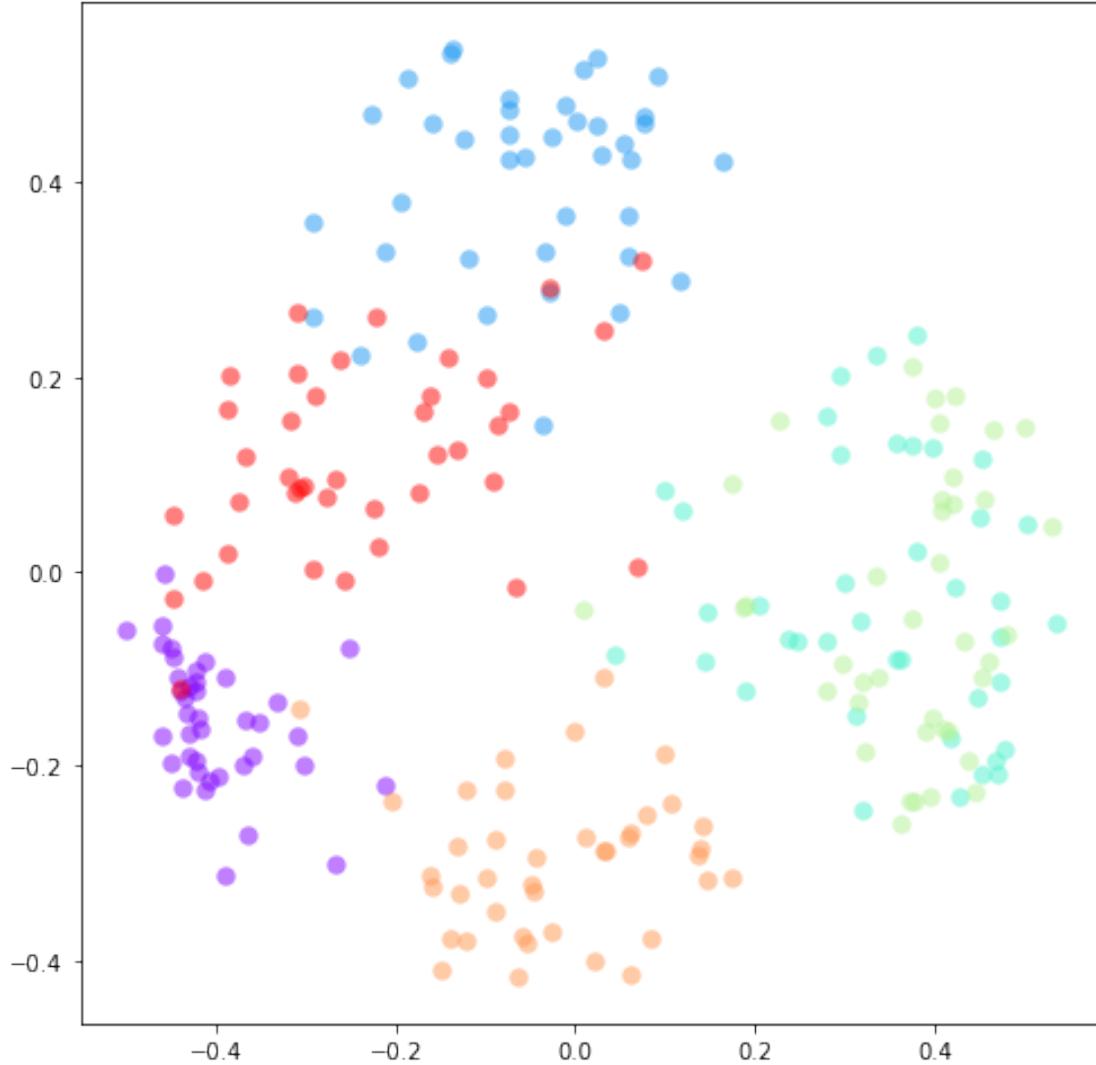
images_ = images.astype("float32")
images_ = tf.convert_to_tensor(images_)

features_ = model(images_)
features = features_.numpy()

print("features.shape = ", features.shape)
```

```
plot_projection_one(
    features = features,
    labels = labels,
    title = "The 2D projection of baseline MobileNet model",
    legend_location = "best",
    savename = "fig_2d_proj_baseline_mobilenet.pdf",
    algo = project_algo
)
del model
```

```
features.shape = (240, 154)
projections.shape = (240, 2)
saved to: figures/fig_2d_proj_baseline_mobilenet.pdf
```



## 41 Projecting with contrastive simple model

```
[14]: model = keras.models.load_model("models/contrastive_simple/main.saved_model")

images_ = images.astype("float32")
images_ = tf.convert_to_tensor(images_)

features_ = model(images_)
features = features_.numpy()

print("features.shape = ", features.shape)
```

```

plot_projection_one(
    features = features,
    labels = labels,

    title = "The 2D projection of contrastive simple model",

    legend_location = "best",

    savename = "fig_2d_proj_contrastive_simple.pdf",
    algo = project_algo
)

del model

```

WARNING:absl:Importing a function  
`(__inference_projhead_output_layer_call_and_return_conditional_losses_28597044)`  
with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function `(__inference_projhead_hidden_1_layer_call_and_return_conditional_losses_28595399)` with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function `(__inference_projhead_hidden_2_layer_call_and_return_conditional_losses_28595421)` with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function `(__inference_projhead_hidden_1_layer_call_and_return_conditional_losses_28597909)` with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function `(__inference__wrapped_model_28593803)` with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function  
`(__inference_projhead_output_layer_call_and_return_conditional_losses_28597080)`  
with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function `(__inference_projhead_hidden_2_layer_call_and_return_conditional_losses_28597934)` with ops with custom gradients. Will likely fail if a gradient is requested.

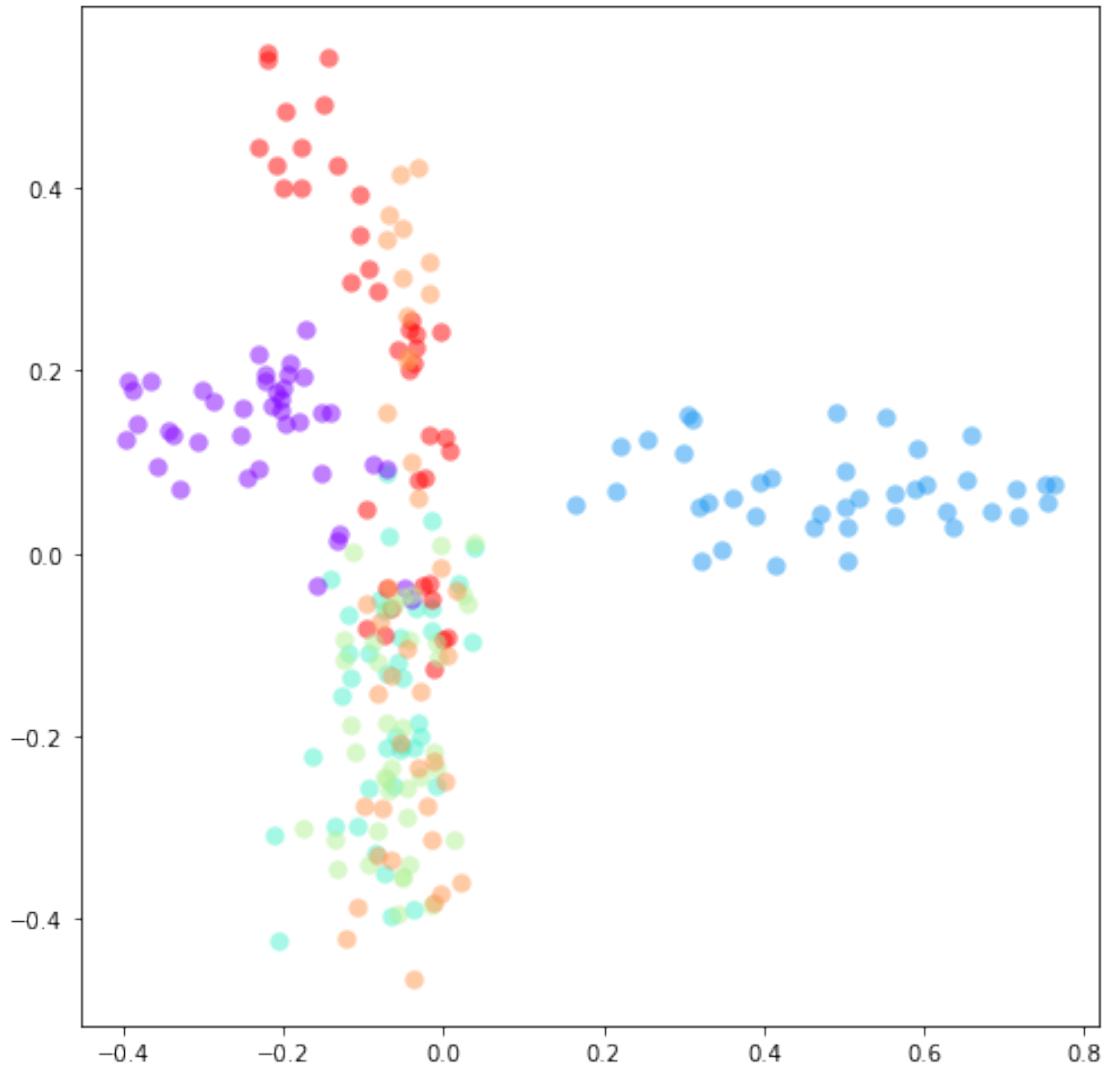
WARNING:absl:Importing a function `(__inference_contrastive_model_17_layer_call_and_return_conditional_losses_28596758)` with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function `(__inference_contrastive_model_17_layer_call_and_return_conditional_losses_28596552)` with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function `(__inference_contrastive_model_17_layer_call_and_return_conditional_losses_28596442)` with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function `(__inference_contrastive_model_17_layer_call_and_return_conditional_losses_28596648)` with ops with custom gradients. Will likely fail if a gradient is requested.

```
features.shape = (240, 512)
projections.shape = (240, 2)
saved to: figures/fig_2d_proj_contrastive_simple.pdf
```



## 42 Projecting with contrastive simple model (temperature scheduler)

```
[29]: model = keras.models.load_model("models/contrastive_simple_dynamic/main.  
→saved_model")
```

```
images_ = images.astype("float32")
images_ = tf.convert_to_tensor(images_)
```

```

features_ = model(images_)
features = features_.numpy()

print("features.shape = ", features.shape)

plot_projection_one(
    features = features,
    labels = labels,

    title = "The 2D projection of contrastive simple model with
    ↴temperature scheduler",

    legend_location = "best",

    savename = "fig_2d_proj_contrastive_simple_dynamic.pdf",
    algo = project_algo
)

del model

```

WARNING:absl:Importing a function  
`(__inference_projhead_hidden_2_layer_call_and_return_conditional_losses_44785)`  
with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function  
`(__inference_contrastive_model_layer_call_and_return_conditional_losses_45816)`  
with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function `(__inference__wrapped_model_43119)` with ops  
with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function  
`(__inference_projhead_output_layer_call_and_return_conditional_losses_46410)`  
with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function  
`(__inference_projhead_hidden_1_layer_call_and_return_conditional_losses_47288)`  
with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function  
`(__inference_projhead_hidden_1_layer_call_and_return_conditional_losses_44763)`  
with ops with custom gradients. Will likely fail if a gradient is requested.

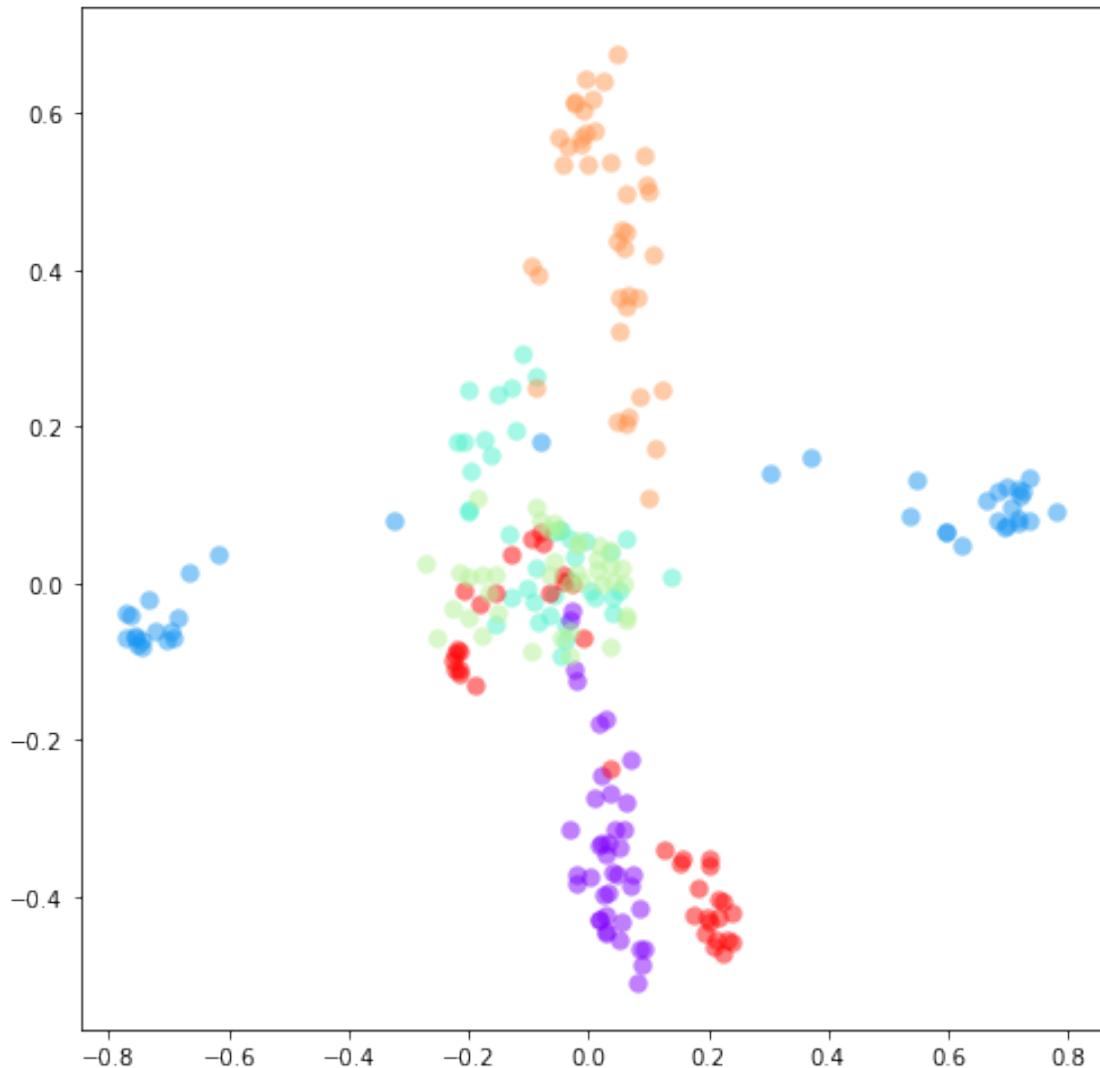
WARNING:absl:Importing a function  
`(__inference_contrastive_model_layer_call_and_return_conditional_losses_45926)`  
with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function  
`(__inference_projhead_output_layer_call_and_return_conditional_losses_46374)`  
with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function  
`(__inference_contrastive_model_layer_call_and_return_conditional_losses_45610)`  
with ops with custom gradients. Will likely fail if a gradient is requested.

```
WARNING:absl:Importing a function
(_inference_contrastive_model_layer_call_and_return_conditional_losses_45720)
with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(_inference_projhead_hidden_2_layer_call_and_return_conditional_losses_47313)
with ops with custom gradients. Will likely fail if a gradient is requested.

features.shape =  (240, 512)
saved to:  figures/fig_2d_proj_contrastive_simple_dynamic.pdf
```



## 43 Projecting with contrastive VGG16

```
[15]: model = keras.models.load_model("models/contrastive_vgg16/main.saved_model")

images_ = images.astype("float32")
images_ = tf.convert_to_tensor(images_)

features_ = model(images_)
features = features_.numpy()

print("features.shape = ", features.shape)

plot_projection_one(
    features = features,
    labels = labels,

    title = "The 2D projection of contrastive VGG16 model",
    legend_location = "best",

    savename = "fig_2d_proj_contrastive_vgg16.pdf",
    algo = project_algo
)

del model
```

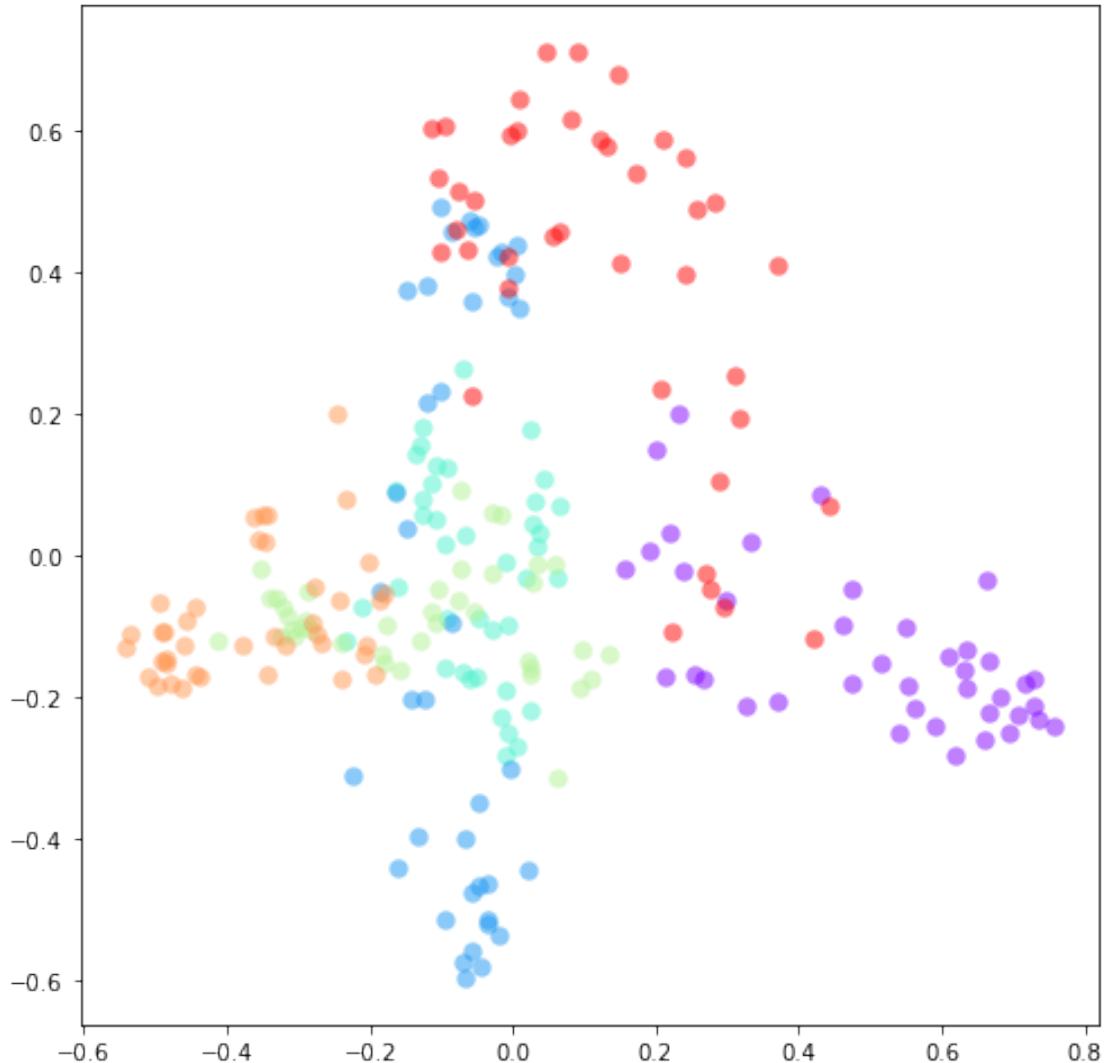
WARNING:absl:Importing a function (`__inference_projhead_hidden_1_layer_call_and_return_conditional_losses_28646246`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_contrastive_model_18_layer_call_and_return_conditional_losses_28645444`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_projhead_output_layer_call_and_return_conditional_losses_28645550`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_contrastive_model_18_layer_call_and_return_conditional_losses_28645042`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_projhead_hidden_1_layer_call_and_return_conditional_losses_28643701`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_contrastive_model_18_layer_call_and_return_conditional_losses_28645176`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_projhead_hidden_2_layer_call_and_return_conditional_losses_28646271`) with ops with custom gradients. Will likely

```

fail if a gradient is requested.
WARNING:absl:Importing a function (<__inference_projhead_hidden_2_layer_call_and_
return_conditional_losses_28643723>) with ops with custom gradients. Will likely
fail if a gradient is requested.
WARNING:absl:Importing a function (<__inference_contrastive_model_18_layer_call_a
nd_return_conditional_losses_28645310>) with ops with custom gradients. Will
likely fail if a gradient is requested.
WARNING:absl:Importing a function (<__inference_projhead_output_layer_call_and_return_conditional_losses_28645514>
with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (<__inference__wrapped_model_28642081>) with ops
with custom gradients. Will likely fail if a gradient is requested.

features.shape = (240, 512)
projections.shape = (240, 2)
saved to: figures/fig_2d_proj_contrastive_vgg16.pdf

```



## 44 Projecting with contrastive ViT

```
[16]: model = keras.models.load_model("models/contrastive_vit/main.saved_model")

images_ = images.astype("float32")
images_ = tf.convert_to_tensor(images_)

features_ = model(images_)
features = features_.numpy()

print("features.shape = ", features.shape)

plot_projection_one(
    features = features,
    labels = labels,

    title = "The 2D projection of contrastive ViT model",
    legend_location = "best",

    savename = "fig_2d_proj_contrastive_vit.pdf",
    algo = project_algo
)

del model
```

WARNING:absl:Importing a function (`__inference_projhead_hidden_2_layer_call_and_return_conditional_losses_28707845`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_contrastive_model_19_layer_call_and_return_conditional_losses_28711287`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_projhead_output_layer_call_and_return_conditional_losses_28713155`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_dense_66_layer_call_and_return_conditional_losses_28706061`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_dense_65_layer_call_and_return_conditional_losses_28713910`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_dense_67_layer_call_and_return_conditional_losses_28714234`) with

```
ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_dense_65_layer_call_and_return_conditional_losses_28705887) with  
ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (__inference__wrapped_model_28704921) with ops  
with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_dense_68_layer_call_and_return_conditional_losses_28706277) with  
ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_dense_69_layer_call_and_return_conditional_losses_28714558) with  
ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_dense_64_layer_call_and_return_conditional_losses_28705845) with  
ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_dense_69_layer_call_and_return_conditional_losses_28706319) with  
ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_dense_66_layer_call_and_return_conditional_losses_28714189) with  
ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (__inference_contrastive_model_19_layer_call_a  
nd_return_conditional_losses_28710785) with ops with custom gradients. Will  
likely fail if a gradient is requested.  
WARNING:absl:Importing a function (__inference_contrastive_model_19_layer_call_a  
nd_return_conditional_losses_28711838) with ops with custom gradients. Will  
likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_dense_64_layer_call_and_return_conditional_losses_28713865) with  
ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (__inference_projhead_hidden_2_layer_call_and_  
return_conditional_losses_28714727) with ops with custom gradients. Will likely  
fail if a gradient is requested.  
WARNING:absl:Importing a function (__inference_contrastive_model_19_layer_call_a  
nd_return_conditional_losses_28710234) with ops with custom gradients. Will  
likely fail if a gradient is requested.  
WARNING:absl:Importing a function (__inference_projhead_hidden_1_layer_call_and_  
return_conditional_losses_28714702) with ops with custom gradients. Will likely  
fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_projhead_output_layer_call_and_return_conditional_losses_28713119)  
with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_model_1_layer_call_and_return_conditional_losses_28712530) with ops  
with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_model_1_layer_call_and_return_conditional_losses_28713049) with ops  
with custom gradients. Will likely fail if a gradient is requested.
```

```
WARNING:absl:Importing a function (__inference_projhead_hidden_1_layer_call_and_return_conditional_losses_28707823) with ops with custom gradients. Will likely fail if a gradient is requested.
```

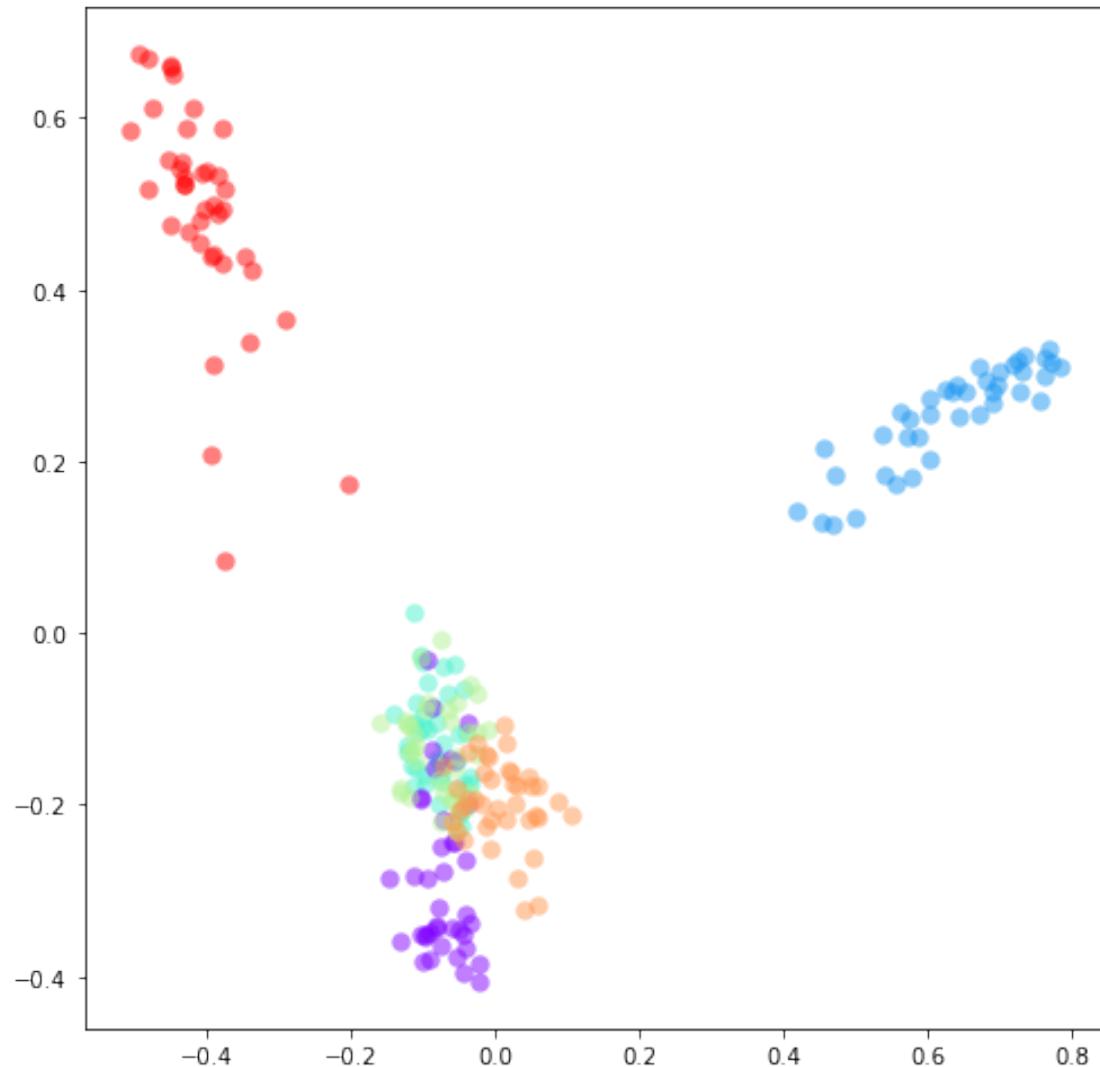
```
WARNING:absl:Importing a function (__inference_dense_68_layer_call_and_return_conditional_losses_28714513) with ops with custom gradients. Will likely fail if a gradient is requested.
```

```
WARNING:absl:Importing a function (__inference_dense_67_layer_call_and_return_conditional_losses_28706103) with ops with custom gradients. Will likely fail if a gradient is requested.
```

```
features.shape = (240, 512)
```

```
projections.shape = (240, 2)
```

```
saved to: figures/fig_2d_proj_contrastive_vit.pdf
```



## 45 The facility of images to features

```
[33]: def images_to_features(images, model_path):
    model = keras.models.load_model(model_path)

    images_ = images.copy().astype("float32")
    images_ = tf.convert_to_tensor(images_)

    features_ = model(images_)
    features = features_.numpy()

    return features
```

## 46 Projecting temperature scheduling

```
[35]: keys = [
    "contrastive-simple-static",
    "contrastive-simple-dynamic",
]

subtitles = keys

mode_paths_dict = {
    "contrastive-simple-static": "models/contrastive_simple/main.saved_model",
    "contrastive-simple-dynamic": "models/contrastive_simple_dynamic/main.
    →saved_model",
}

features_dict = {
    key: images_to_features(images, mode_paths_dict[key])
    for key in keys
}
```

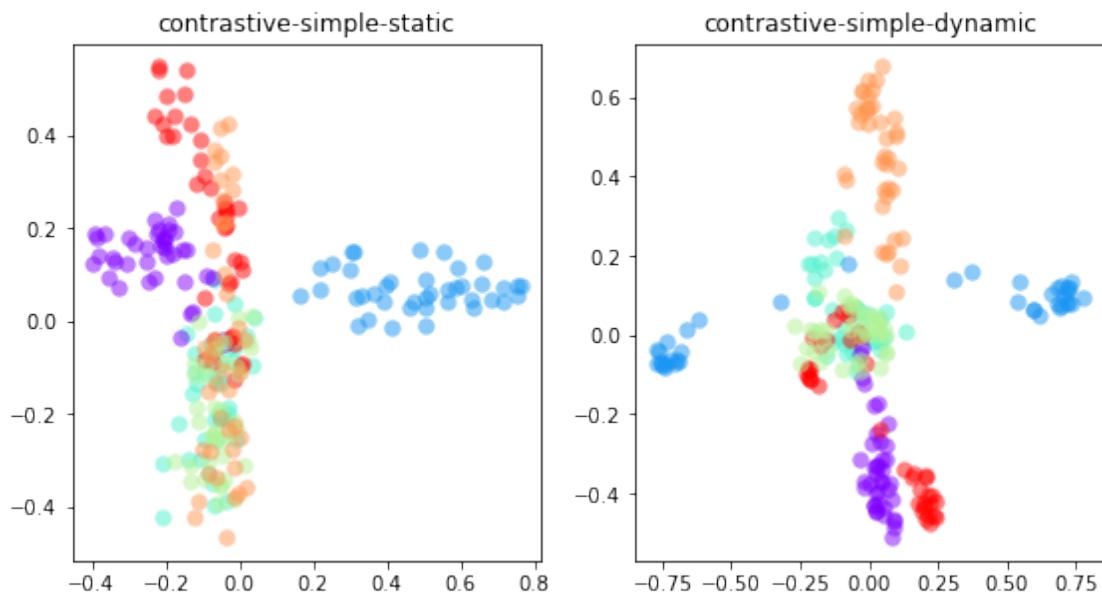
```
WARNING:absl:Importing a function
(_inference_projhead_output_layer_call_and_return_conditional_losses_28597044)
with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (_inference_projhead_hidden_1_layer_call_and_
return_conditional_losses_28595399) with ops with custom gradients. Will likely
fail if a gradient is requested.
WARNING:absl:Importing a function (_inference_projhead_hidden_2_layer_call_and_
return_conditional_losses_28595421) with ops with custom gradients. Will likely
fail if a gradient is requested.
WARNING:absl:Importing a function (_inference_projhead_hidden_1_layer_call_and_
return_conditional_losses_28597909) with ops with custom gradients. Will likely
fail if a gradient is requested.
WARNING:absl:Importing a function (_inference__wrapped_model_28593803) with ops
```

with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_\_inference\_projhead\_output\_layer\_call\_and\_return\_conditional\_losses\_28597080)  
with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (\_\_inference\_projhead\_hidden\_2\_layer\_call\_and\_return\_conditional\_losses\_28597934) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (\_\_inference\_contrastive\_model\_17\_layer\_call\_and\_return\_conditional\_losses\_28596758) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (\_\_inference\_contrastive\_model\_17\_layer\_call\_and\_return\_conditional\_losses\_28596552) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (\_\_inference\_contrastive\_model\_17\_layer\_call\_and\_return\_conditional\_losses\_28596442) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (\_\_inference\_contrastive\_model\_17\_layer\_call\_and\_return\_conditional\_losses\_28596648) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_\_inference\_projhead\_hidden\_2\_layer\_call\_and\_return\_conditional\_losses\_44785)  
with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_\_inference\_contrastive\_model\_layer\_call\_and\_return\_conditional\_losses\_45816)  
with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_\_inference\_\_wrapped\_model\_43119) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_\_inference\_projhead\_output\_layer\_call\_and\_return\_conditional\_losses\_46410)  
with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_\_inference\_projhead\_hidden\_1\_layer\_call\_and\_return\_conditional\_losses\_47288)  
with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_\_inference\_projhead\_hidden\_1\_layer\_call\_and\_return\_conditional\_losses\_44763)  
with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_\_inference\_contrastive\_model\_layer\_call\_and\_return\_conditional\_losses\_45926)  
with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_\_inference\_projhead\_output\_layer\_call\_and\_return\_conditional\_losses\_46374)  
with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_\_inference\_contrastive\_model\_layer\_call\_and\_return\_conditional\_losses\_45610)  
with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_\_inference\_contrastive\_model\_layer\_call\_and\_return\_conditional\_losses\_45720)  
with ops with custom gradients. Will likely fail if a gradient is requested.

```
WARNING:absl:Importing a function
(_inference_projhead_hidden_2_layer_call_and_return_conditional_losses_47313)
with ops with custom gradients. Will likely fail if a gradient is requested.
```

```
[38]: plot_projections(
    features_dict = features_dict,
    labels = labels,
    keys = keys,
    subtitles = subtitles,
    title = "Feature projection comparison w.r.t. static and
    ↪dynamic temperature",
    ncols = 2,
    savename = "fig_projection_comparison_temperature.pdf",
    #kernelpca-cosine, tsne-cosine, umap-cosine
    algo = "kernelpca-cosine",
    subfig_height = 4,
    subfig_width = 4,
)
```

nrows = 1 ncols = 2  
saved to: figures/fig\_projection\_comparison\_temperature.pdf



## 47 Projecting model comparison

```
[7]: keys = [
    "raw-image",
    "baseline-simple",
    "baseline-vgg16",
    "baseline-resnet",
    "baseline-mobilenet",
    "contrastive-simple",
    "contrastive-vgg16",
    "contrastive-vit"
]

subtitles = keys

mode_paths_dict = {
    "baseline-simple": "models/baseline_simple/main.saved_model",
    "baseline-vgg16": "models/baseline_vgg16/main.saved_model",
    "baseline-resnet": "models/baseline_resnet/main.saved_model",
    "baseline-mobilenet": "models/baseline_mobilenet/main.saved_model",

    "contrastive-simple": "models/contrastive_simple/main.saved_model",
    "contrastive-vgg16": "models/contrastive_vgg16/main.saved_model",
    "contrastive-vit": "models/contrastive_vit/main.saved_model",
}

features_dict = {
    key: images_to_features(images, mode_paths_dict[key])
    for key in keys if key != "raw-image"
}

n_feature_dims = np.prod(images[0].shape)
features = images.copy().reshape(-1, n_feature_dims)
features_dict["raw-image"] = features
```

WARNING:absl:Importing a function  
(\_inference\_conv2d\_1\_layer\_call\_and\_return\_conditional\_losses\_28219846) with  
ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (\_inference\_baseline\_simple\_model\_layer\_call\_  
and\_return\_conditional\_losses\_28221690) with ops with custom gradients. Will  
likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_inference\_encoder\_simple\_layer\_call\_and\_return\_conditional\_losses\_28222766)  
with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(\_inference\_conv2d\_2\_layer\_call\_and\_return\_conditional\_losses\_28223436) with  
ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (\_inference\_wrapped\_model\_28218849) with ops

with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_baseline_simple_model_layer_call_and_return_conditional_losses_28222148`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(`__inference_conv2d_3_layer_call_and_return_conditional_losses_28223461`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(`__inference_conv2d_2_layer_call_and_return_conditional_losses_28219902`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(`__inference_conv2d_layer_call_and_return_conditional_losses_28223235`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(`__inference_conv2d_1_layer_call_and_return_conditional_losses_28223260`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(`__inference_conv2d_layer_call_and_return_conditional_losses_28219824`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(`__inference_encoder_simple_layer_call_and_return_conditional_losses_28222668`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(`__inference_conv2d_3_layer_call_and_return_conditional_losses_28219924`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(`__inference_projhead_output_layer_call_and_return_conditional_losses_28597044`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_projhead_hidden_1_layer_call_and_return_conditional_losses_28595399`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_projhead_hidden_2_layer_call_and_return_conditional_losses_28595421`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_projhead_hidden_1_layer_call_and_return_conditional_losses_28597909`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference__wrapped_model_28593803`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(`__inference_projhead_output_layer_call_and_return_conditional_losses_28597080`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_projhead_hidden_2_layer_call_and_return_conditional_losses_28597934`) with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (`__inference_contrastive_model_17_layer_call_and_return_conditional_losses_28596758`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference_contrastive_model_17_layer_call_and_return_conditional_losses_28596552`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference_contrastive_model_17_layer_call_and_return_conditional_losses_28596442`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference_contrastive_model_17_layer_call_and_return_conditional_losses_28596648`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference_projhead_hidden_1_layer_call_and_return_conditional_losses_28646246`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference_contrastive_model_18_layer_call_and_return_conditional_losses_28645444`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference_projhead_output_layer_call_and_return_conditional_losses_28645550`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference_contrastive_model_18_layer_call_and_return_conditional_losses_28645042`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference_projhead_hidden_1_layer_call_and_return_conditional_losses_28643701`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference_contrastive_model_18_layer_call_and_return_conditional_losses_28645176`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference_projhead_hidden_2_layer_call_and_return_conditional_losses_28646271`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference_projhead_hidden_2_layer_call_and_return_conditional_losses_28643723`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference_contrastive_model_18_layer_call_and_return_conditional_losses_28645310`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference_projhead_output_layer_call_and_return_conditional_losses_28645514`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference__wrapped_model_28642081`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function (`__inference_projhead_hidden_2_layer_call_and_return_conditional_losses_28707845`) with ops with custom gradients. Will likely fail if a gradient is requested.

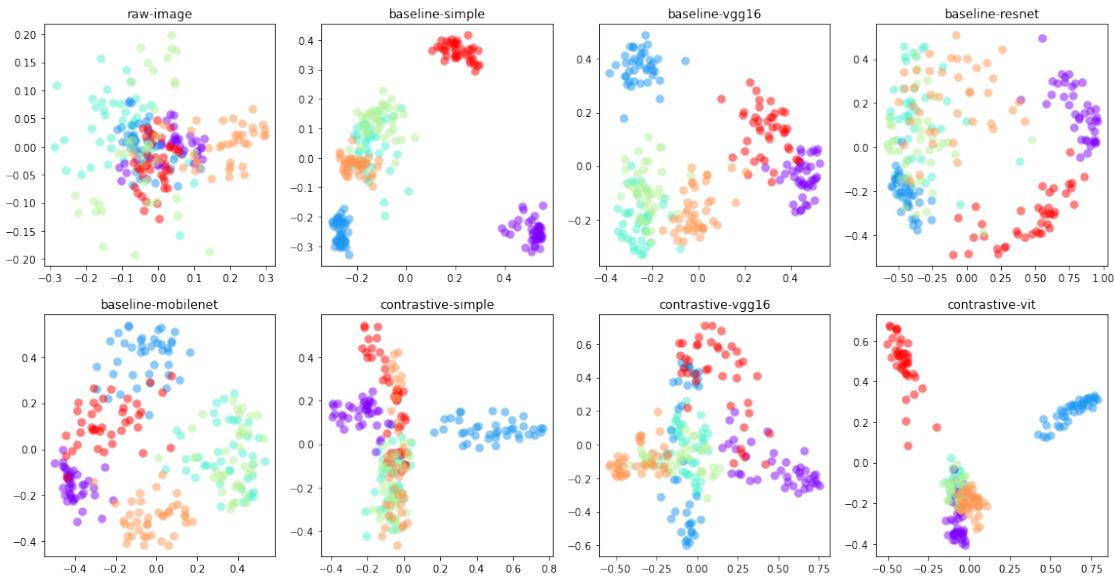
WARNING:absl:Importing a function (`__inference_contrastive_model_19_layer_call_and_return_conditional_losses_28711287`) with ops with custom gradients. Will likely fail if a gradient is requested.

WARNING:absl:Importing a function

```
(__inference_projhead_output_layer_call_and_return_conditional_losses_28713155)
with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_dense_66_layer_call_and_return_conditional_losses_28706061) with
ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_dense_65_layer_call_and_return_conditional_losses_28713910) with
ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_dense_67_layer_call_and_return_conditional_losses_28714234) with
ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_dense_65_layer_call_and_return_conditional_losses_28705887) with
ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference__wrapped_model_28704921) with ops
with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_dense_68_layer_call_and_return_conditional_losses_28706277) with
ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_dense_69_layer_call_and_return_conditional_losses_28714558) with
ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_dense_64_layer_call_and_return_conditional_losses_28705845) with
ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_dense_69_layer_call_and_return_conditional_losses_28706319) with
ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_dense_66_layer_call_and_return_conditional_losses_28714189) with
ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_contrastive_model_19_layer_call_a
nd_return_conditional_losses_28710785) with ops with custom gradients. Will
likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_contrastive_model_19_layer_call_a
nd_return_conditional_losses_28711838) with ops with custom gradients. Will
likely fail if a gradient is requested.
WARNING:absl:Importing a function
(__inference_dense_64_layer_call_and_return_conditional_losses_28713865) with
ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_projhead_hidden_2_layer_call_and_
return_conditional_losses_28714727) with ops with custom gradients. Will likely
fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_contrastive_model_19_layer_call_a
nd_return_conditional_losses_28710234) with ops with custom gradients. Will
likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_projhead_hidden_1_layer_call_and_
return_conditional_losses_28714702) with ops with custom gradients. Will likely
```

```
fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_projhead_output_layer_call_and_return_conditional_losses_28713119)  
with ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_model_1_layer_call_and_return_conditional_losses_28712530) with ops  
with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_model_1_layer_call_and_return_conditional_losses_28713049) with ops  
with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function (__inference_projhead_hidden_1_layer_call_and_  
return_conditional_losses_28707823) with ops with custom gradients. Will likely  
fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_dense_68_layer_call_and_return_conditional_losses_28714513) with  
ops with custom gradients. Will likely fail if a gradient is requested.  
WARNING:absl:Importing a function  
(__inference_dense_67_layer_call_and_return_conditional_losses_28706103) with  
ops with custom gradients. Will likely fail if a gradient is requested.
```

```
[13]: plot_projections(  
                      features_dict = features_dict,  
                      labels = labels,  
  
                      keys = keys,  
  
                      subtitles = subtitles,  
                      title = "Feature projection comparison",  
  
                      ncols = 4,  
  
                      savename = "fig_projection_comparison.pdf",  
  
                      #kernel pca-cosine, tsne-cosine, umap-cosine  
                      algo = "kernel pca-cosine",  
  
                      subfig_height = 4,  
                      subfig_width = 4,  
                      )  
  
nrows = 2 ncols = 4  
saved to: figures/fig_projection_comparison.pdf
```



## 48 Conclusions

[ ]: