Chapter 16

Deep Learning Classification Based on Raw MRI Images

Sebastian Moguilner and Agustin Ibañez

Abstract

In this chapter, we describe a step-by-step implementation of an automated anatomical MRI feature extractor based on artificial intelligence machine learning for classification. We applied the *DenseNet*—a state-of-the-art convolutional neural network producing more robust results than previous deep learning network architectures—to data from male ($n = 400$) and female ($n = 400$), age-, and education- matched healthy adult subjects. Moreover, we illustrate how an occlusion sensitivity analysis provides meaningful insights about the relevant information that the neural network used to make accurate classifications. This addresses the "black-box" limitations inherent in many deep learning implementations. The use of this approach with a specific dataset demonstrates how future implementations can use raw MRI scans to study a range of outcome measures, including neurological and psychiatric disorders.

Key words Deep Learning, MRI, Artificial intelligence, Convolutional neural network, Occlusion sensitivity analysis

1 Introduction

Medical imaging computer-aided diagnosis has been a field of intense research over the last decades. Its primary motivation is to reduce diagnostic errors, automatize procedures with large datasets, and provide additional insights to better interpret the images. However, research translation from the lab to the real world (i.e., from "bench to bedside") has often been limited by a failure to generalize to novel datasets. In particular, rule-based algorithms lack the flexibility needed to handle heterogeneous samples. Current developments in computer vision, stemming from the field of artificial intelligence [see Glossary], provide an innovative solution to create flexible decision-making pipelines automatically. Moreover, thanks to recent improvements in dedicated computing hardware, it is now possible to handle large neuroimaging datasets.

Deep learning [see Glossary] computer vision methods, based on convolutional neural networks (CNN) [see Glossary], are characterized by their flexibility in evaluating images without prior orientation, metric, or shape conventions that are usually set prior

Robert Whelan and Hervé Lemaître (eds.), *Methods for Analyzing Large Neuroimaging Datasets*, Neuromethods, vol. 218, https://doi.org/10.1007/978-1-0716-4260-3_16,

image processing [1]. In this way, possible biases in the preprocessing steps such as image filtering, segmentation, rotation, and smoothing are eliminated because the input consists of unprocessed (raw) data, from which the most relevant image features are automatically extracted [2]. This approach has already proven successful in image recognition in general [3], in medical radiology [4], and neuroradiological domains such as in dementia characterization [5, 6]. However, open procedures with step-by-step detailed examples on how to implement a Deep Learning neural network on open access data with interpretable results are scarce.

This chapter will describe the implementation of a state-of-the-art deep learning algorithm called the *DenseNet* [7]. We show a step-by-step example to use the DenseNet on brain anatomical MRI images to classify male ($n = 400$) and female (n-400), age- and education- matched healthy adult subjects. Sex differences in the human brain are of great interest for studies of neuropsychological traits [8] and the differential prevalence of psychiatric [9] and neurological disorders such as Alzheimer's disease [10]. Despite its high relevance in epidemiological research, biological sex differences have been defined as an under-explored and often controversial subject in the neuroscientific literature [11, 12]. Crucially, previous studies looking at sex-related brain differences suffered from low statistical power due to small sample sizes, having a mean sample size of 130 participants [13]. This context calls for new and unbiased approaches able to handle more extensive databases, in which computer-vision algorithms with automatic feature extraction [see Glossary] on big-data contexts may help to gain further insights on brain differences that are undetectable by the human eye and/or by traditional machine learning [see Glossary] approaches.

2 Methods

2.1 The DenseNet Deep Learning Algorithm

The DenseNet is a state-of-the-art deep learning CNN employed in computer vision tasks aided by artificial intelligence. In this network architecture, each layer is connected to every other layer in a feed-forward [see Glossary] fashion, producing networks that are substantially deeper, more accurate, and more efficient to train (see Fig. 1c for the DenseNet diagram).

Consider $a^{[0]}$ as the input volume passed through a CNN, with L as the number of layers in the network, and g the non-linear transformation of lth layer. Traditional feed-forward convolutional networks connect the output of the lth layer as input to the $(l + 1)$th layer, giving the following transition layer: $a^{[l]} = g(a^{[l-1]})$. Other CNN architectures such as the ResNet [3] bypasses the non-linear transformations with an identity function:

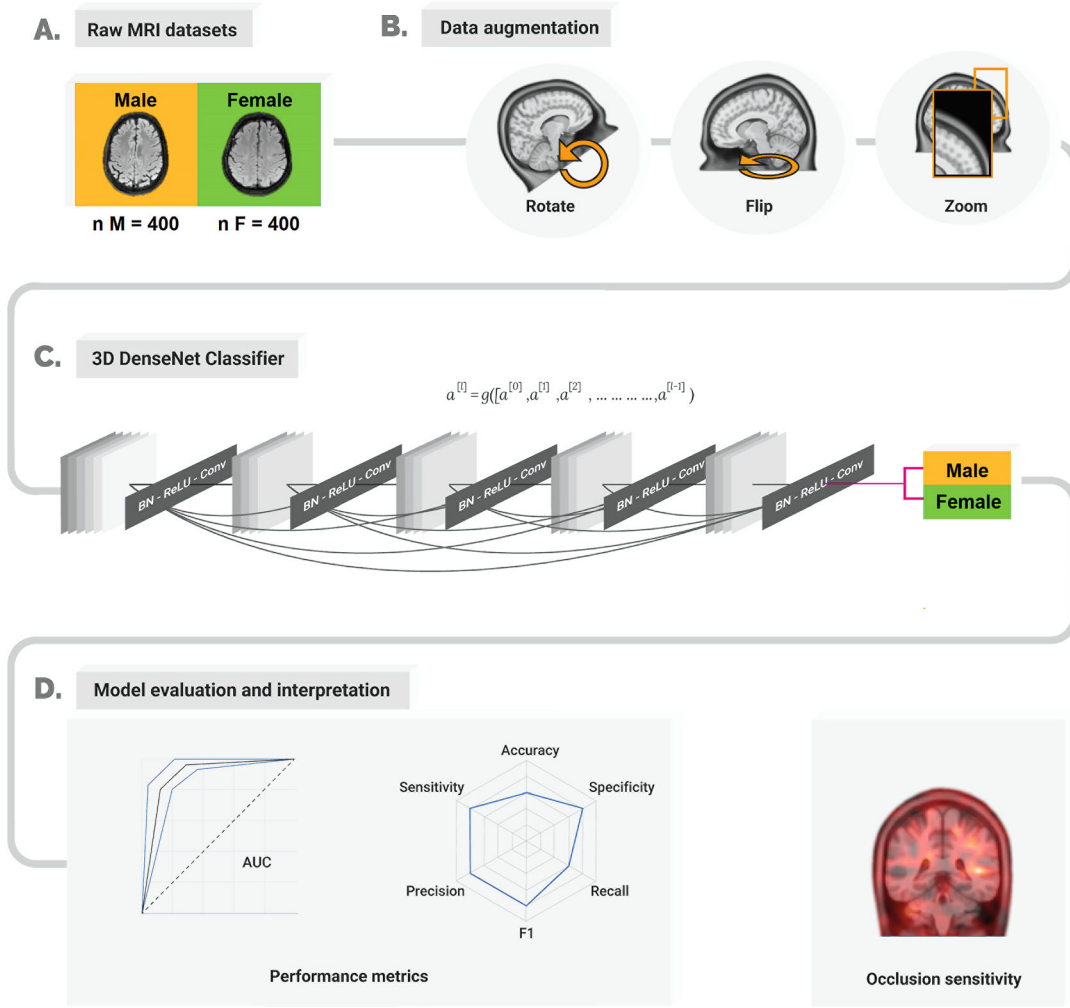

Fig. 1 Step-by-step pipeline. (**a**) Number of raw MRI datasets employed in the analysis (before data augmentation) for male and female groups and related demographic matching. (**b**) Data preparation and augmentation pipeline consisting of random volume rotations, flipping, zooms and an image size scaling. The random augmentation process increased the sample size by a factor of 10. (**c**) 3D DenseNet network architecture consisting of a sequence of Dense Blocks and transition layers consisting of a Batch Normalization (BN), a rectified linear unit (ReLU), and a convolution transformation, ending in a prediction layer to produce the output. (**d**) Model evaluation interpretation, with the performance metrics consisting of the ROC [see Glossary] curve and an AUC [see Glossary] report, a radar plot showing the accuracy, sensitivity, specificity, precision, recall, and F1 metrics. An *occlusion sensitivity analysis* [see Glossary] was developed to obtain the most relevant image information used for the classification

$$a^{[l]} = g\left(a^{[l-1]}\right) + a^{[l-1]} \quad (1)$$

In this way, the gradient can flow directly through the identity function, from early layers to the subsequent layers. Unlike the ResNet, the DenseNet does not sum the output feature maps from preceding layers, but concatenates them instead in the following way for the lth layer:

$$a^{[l]} = g\left(\left[a^{[0]}, a^{[1]}, a^{[2]}, \ldots, a^{[l-1]}\right]\right) \quad (2)$$

where ($[a^{[0]}, a^{[1]}, a^{[2]}, \ldots, a^{[l-1]}]$) is the concatenation of output feature maps of the preceding layers. Since this feature map grouping requires the feature maps to have equal dimensions, the DenseNet is divided into Dense Blocks, in which the dimensions of the feature maps remain constant within a block, but the number of filters change in the transition layers between them. For each transition layer, a Batch Normalization (BN), a rectified linear unit (ReLU), and a convolution followed by average pooling is applied. Another important feature of DenseNet is the growth rate, defined for the

$$l\text{th feature map as } k_0 + k \times (l-1),$$

with k_0 the number of feature maps at the input layer, and k the growth rate. Compared to traditional CNNs, the DenseNet has several advantages: It further reduces the vanishing-gradient problem, strengthens feature propagation, encourages feature reuse, and substantially reduces the number parameters when compared to other neural networks, thus reducing overfitting and producing more robust results [14].

2.2 Software and Coding

To run the code of this chapter, several library dependencies should be installed. However, all of them are contained in the MONAI PyTorch-based API (https://github.com/Project-MONAI). Project Medical Open Network for AI (MONAI) is an initiative to share and develop best practices for AI in healthcare imaging across academia and enterprise researchers. This collaboration has expanded to include academic and industry leaders throughout the medical imaging field. Project MONAI has released multiple open-source PyTorch-based frameworks for annotating, building, training, deploying, and optimizing AI workflows in healthcare. These standardization frameworks provide high-quality, user-friendly software that facilitates reproducibility and easy integration. The suite of libraries, tools, and SDKs within MONAI provides a robust and common foundation that covers the end-to-end medical AI life cycle, from annotation through deployment.

2.3 Requirements and Setup

The current example requires the installation of the matplotlib library and the Jupyter Notebook GUI. These can be installed with:

python -m pip install -U pip

python -m pip install -U matplotlib

python -m pip install -U notebook

> **Note**
> To run the notebook from Google Colab, a GPU setup is needed. To use GPU resources through Colab, please remember to change the runtime type to GPU:
>
> 1. *From the Runtime menu select Change runtime type*
> 2. *Choose GPU from the drop-down menu*
>
> Click SAVE This will reset the notebook and may ask you if you are a robot (these instructions assume you are not).

Running:

nvidia-smi

in a cell, it will verify this has worked and show you what kind of hardware you have access to.

To install the current MONAI milestone release:

pip install monai

To install the weekly preview release:

pip install monai-weekly

The packages installed using pip install could be removed by:

pip uninstall -y monai

pip uninstall -y monai-weekly

From conda-forge, to install the current milestone release:

conda install -c conda-forge monai

You can verify the installation by:

python -c 'import monai; monai.config.print_config()'

If the installation is successful, this command will print out the MONAI version information.

2.4 Dataset

The open dataset included in this example comprises of healthy subjects' (Males $n = 400$, Females $n = 400$) T1-weighted data, compliant with the Brain Imaging Data Structure (BIDS), was

Table 1
Demographic statistical results for the database

	Female $n = 400$	Male $n = 400$	Statistics
Age (years)	22.86 (1.72)	22.85 (1.67)	$F = 0.08$ $p = 0.92$[a], $d = 0.006$
Education level (low/medium/high)	40:184:176	42:176:182	$\chi^2 = 0.32$, $p = 0.84$[b]

Results are presented as mean (SD). Age data was assessed through independent two-sample t test. Level of education analyzed via Pearson's chi-squared (χ^2) test. Effects sizes were calculated through Cohen's d (d)
[a] *p*-values calculated via independent two-sample t-test
[b] *p*-values calculated via chi-squared test ($\chi 2$)

downloaded from the Amsterdam Open MRI Collection (AOMIC) (https://openneuro.org/datasets/ds003097/versions/1.2.1) [see Chapter 2]. The samples were matched on age and education level (Table 1). This dataset was collected between 2010 and 2012. The faculty's ethical committee of the University of Amsterdam approved this study before data collection started (EC number: 2010-BC-1345). Prior to the experiment, subjects were informed about the goal and scope of the research, the MRI procedure, safety measures, general experimental procedures, privacy and data sharing concerns, and voluntary nature of the project. Before the start of the experiment, subjects signed an informed consent form and were screened for MRI safety.

All MRI structural data was obtained on the same Philips 3 T Intera scanner. At the start of each scan session, a low-resolution survey scan was made, which was used to determine the location of the field-of-view. For all structural (T1-weighted) the slice stack was not angulated, and the following parameters were used: FOV (RL/AP/FH; mm) = 160 × 256 × 256, Voxel size (mm) = 1 × 1 × 1, TR/TE (ms) = 8.1/3.7, Flip angle (deg.) = 8, Acquisition direction = Sagittal, Duration = 5 min 58 s.

3 Step-by-Step Code Script

3.1 Load Libraries and Dependencies

In this section, we begin installing MONAI via the pip package management system and importing the necessary libraries and dependencies. Some of them may not be installed on your system so you should proceed with installing them via the *pip* command.

Script 1. Installation of packages and importing libraries

```
!python -c "import monai" || pip install -q "monai-weekly[nibabel, tqdm]"

import logging
import os
import sys
import tempfile
import shutil

import matplotlib.pyplot as plt
import torch
from torch.utils.tensorboard import SummaryWriter

import numpy as np
import glob
import pandas as pd

import monai
from monai.apps import download_and_extract
from monai.config import print_config
from monai.data import CacheDataset, DataLoader, ImageDataset
from monai.data.meta_obj import MetaObj
from monai.data.utils import decollate_batch
from monai.transforms import EnsureChannelFirst, Compose, RandRotate, Resize,
ScaleIntensity, RandFlip, EnsureType, RandZoom, Activations, AsDiscrete, ToNumpy
from monai.metrics import ROCAUCMetric
from sklearn.metrics import classification_report, confusion_matrix

import matplotlib.tri as tri
from scipy import ndimage

import sklearn.metrics as metrics
from matplotlib.pyplot import figure
import matplotlib.pyplot as plt
import random
from random import sample

logging.basicConfig(stream=sys.stdout, level=logging.INFO)
print_config()
```

3.2 Load the T1-Weighted Images from the Downloaded Database and Extract Labels

Now we begin building a list of the nii.gz anatomical files using the glob library. The path should point to the folder where you downloaded the files. Then we load the demographic information table using Pandas. Finally, we extract only the labels (Male/Female) of the files that have matched demographic variables using a *for* loop.

Script 2. Data loading

```
### Load MRI data and demographic variables

images = glob.glob('/content/drive/MyDrive/DL/**/*1_T1w.nii.gz', recursive=True)
images.sort()

df = pd.read_csv('behavioral_data_matched.csv')

shortlist = []
labels = []

for i in range(len(images)):
  if images[i][29:37] in df['ID'].values:
    shortlist.append(images[i])

labels = df['sex'].values
```

3.3 Split the Training, Validation, and Testing Files

Having the matched image and labels dataset, we proceed to split the dataset into separate training, validation, and testing subsets. This process ensures that we are able to test the generalization of the results by using an independent test dataset. Then we define the data transformations to get a bigger training set. For this purpose, we apply random brain volume rotations (minimum degree = 0°, maximum = 180°), brain volume flipping (inverse mirror-like image), and image zoom (1× to 1.5×) to focus on different brain areas each time.

Script 3. Data split and augmentation settings

```
### Split the dataset into training, validation, and testing sets

imtrain = images[:300]
labtrain = labels[:300]
imval = images[300:400]
labval = labels[300:400]
imtest = images[400:800]
labtest = labels[400:800]

# Define transforms
train_transforms = Compose([ScaleIntensity(), EnsureChannelFirst(), Resize((50, 100, 100))])
val_transforms = Compose([ScaleIntensity(), EnsureChannelFirst(), Resize((50, 100, 100))])
train_ds = ImageDataset(image_files=imtrain, labels=labtrain, transform=train_transforms)
```

```
############ DATA AUGMENTATION
aug_transforms = Compose([ScaleIntensity(), EnsureChannelFirst(), Resize((50, 100,
100)), RandRotate(), RandFlip(), RandZoom()])
aug_ds = ImageDataset(image_files=imtrain, labels=labtrain, transform=aug_transforms)
aug_ds2 = ImageDataset(image_files=imtrain, labels=labtrain, transform=aug_transforms)
aug_ds3 = ImageDataset(image_files=imtrain, labels=labtrain, transform=aug_transforms)
aug_ds4 = ImageDataset(image_files=imtrain, labels=labtrain, transform=aug_transforms)
aug_ds5 = ImageDataset(image_files=imtrain, labels=labtrain, transform=aug_transforms)
aug_ds6 = ImageDataset(image_files=imtrain, labels=labtrain, transform=aug_transforms)
aug_ds7 = ImageDataset(image_files=imtrain, labels=labtrain, transform=aug_transforms)
aug_ds8 = ImageDataset(image_files=imtrain, labels=labtrain, transform=aug_transforms)
aug_ds9 = ImageDataset(image_files=imtrain, labels=labtrain, transform=aug_transforms)
aug_ds10 = ImageDataset(
   image_files=imtrain, labels=labtrain, transform=aug_transforms)
train_ds = torch.utils.data.ConcatDataset([train_ds, aug_ds, aug_ds2, aug_ds3, aug_ds4,
aug_ds5, aug_ds6, aug_ds7, aug_ds8, aug_ds9, aug_ds10])

############
# create a training data loader
train_loader = DataLoader(train_ds, batch_size=5, shuffle=True, num_workers=2,
pin_memory=torch.cuda.is_available())

# create a validation data loader
val_ds = ImageDataset(image_files=imval, labels=labval, transform=val_transforms)
val_loader = DataLoader(val_ds, batch_size=2, num_workers=2,
pin_memory=torch.cuda.is_available())

# create a test data loader
test_ds = ImageDataset(image_files=imtest, labels=labtest, transform=val_transforms)
test_loader = DataLoader(test_ds, batch_size=1, num_workers=2,
pin_memory=torch.cuda.is_available())
```

3.4 Create the DenseNet Model

In this setup, we will setup the DenseNet model 121 (*see* Table 2 for network architecture details). We set the initial learning to $1e^{-5}$, the batch size parameter was set to 50 samples, and the maximum number of epochs to 10. The Adam optimizer [see Glossary] will be used to minimize the Cross Entropy loss of the 3D-DenseNet during training process [15].

Table 2
DenseNet 121 architecture details

Layers	Output size	Transformation
Convolution	112 × 112 × 112	7 × 7 × 7 convolution, stride 2
Pooling	56 × 56 × 56	3 × 3 × 3 max pooling, stride 2
Dense block 1	56 × 56 × 56	Input concatenation
Transition layer 1	56 × 56 × 56 28 × 28 × 28	1 × 1 × 1 convolution 2 × 2 × 2 average pooling, stride 2
Dense block 2	28 × 28 × 28	Input concatenation
Transition layer 2	28 × 28 × 28 14 × 14 × 14	2 × 2 × 2 average pooling, stride 2 1 × 1 × 1 convolution
Dense block 3	14 × 14 × 14	Input concatenation
Transition layer 3	14 × 14 × 14 7 × 7 × 7	2 × 2 × 2 average pooling, stride 2 1 × 1 × 1 convolution
Dense block 4	7 × 7 × 7	Input concatenation
Classification layer	1 × 1 × 1	7 × 7 × 7 global average pooling Fully connected softmax

Script 4. DenseNet model definition

```
# Create DenseNet121, CrossEntropyLoss and Adam optimizer
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

model = monai.networks.nets.DenseNet121(
    spatial_dims=3, in_channels=1, out_channels=2).to(device)
loss_function = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), 1e-5)

# start training
val_interval = 2
best_metric = -1
best_metric_epoch = -1
epoch_loss_values = []
metric_values = []
writer = SummaryWriter()
max_epochs = 10
for epoch in range(max_epochs):
    print("-" * 10)
    print(f"epoch {epoch + 1}/{max_epochs}")
    model.train()
    epoch_loss = 0
    step = 0
```

```
    for batch_data in train_loader:
        step += 1
        inputs, labels = batch_data[0].to(device), batch_data[1].to(device)
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = loss_function(outputs, labels)
        loss.backward()
        optimizer.step()
        epoch_loss += loss.item()
        epoch_len = len(train_ds) // train_loader.batch_size
        print(f"{step}/{epoch_len}, train_loss: {loss.item():.4f}")
        writer.add_scalar("train_loss", loss.item(), epoch_len * epoch + step)
    epoch_loss /= step
    epoch_loss_values.append(epoch_loss)
    print(f"epoch {epoch + 1} average loss: {epoch_loss:.4f}")
    if (epoch + 1) % val_interval == 0:
        model.eval()
        with torch.no_grad():
            num_correct = 0.0
            metric_count = 0
            for val_data in val_loader:
                # val_images, val_labels = val_data["img"].to(
                #     device), val_data["label"].to(device)
                val_images, val_labels = val_data[0].to(device), val_data[1].to(device)
                val_outputs = model(val_images)
                value = torch.eq(val_outputs.argmax(dim=1), val_labels)
                metric_count += len(value)
                num_correct += value.sum().item()
            metric = num_correct / metric_count
            metric_values.append(metric)
            if metric > best_metric:
                best_metric = metric
                best_metric_epoch = epoch + 1
                torch.save(model.state_dict(),
                           "best_metric_model_classification3d_array.pth")
                print("saved new best metric model")
            print(
                "current epoch: {} current accuracy: {:.4f} "
                "best accuracy: {:.4f} at epoch {}".format(
                    epoch + 1, metric, best_metric, best_metric_epoch
                )
            )
            writer.add_scalar("val_accuracy", metric, epoch + 1)
print(
    f"train completed, best_metric: {best_metric:.4f} "
    f"at epoch: {best_metric_epoch}")
writer.close()
```

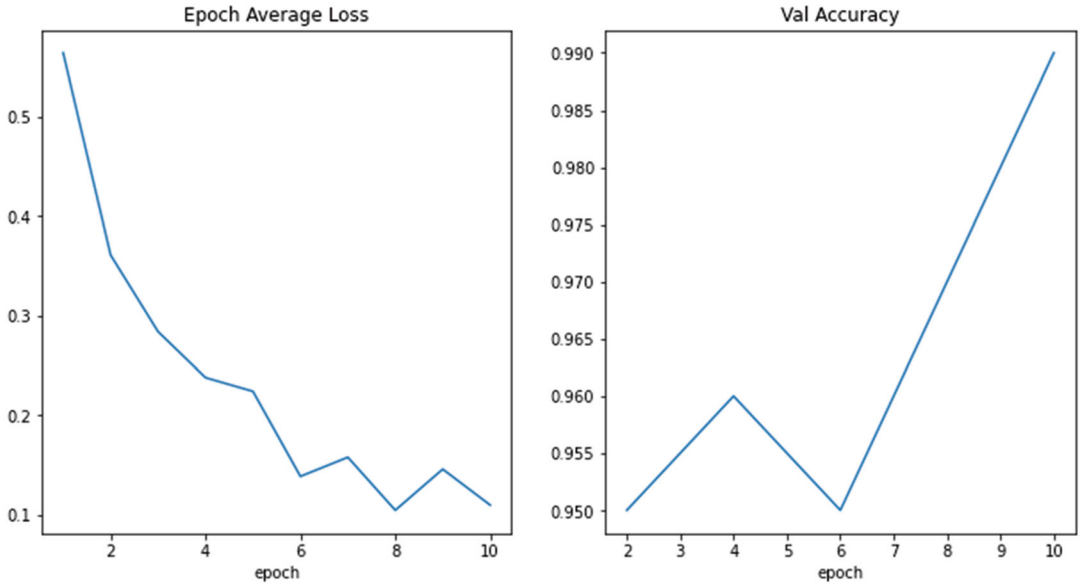

Fig. 2 Epoch average loss, indicating the error across validation runs and validation accuracy plot to check if the network is learning from the input data

3.5 Plot the Average Loss and Validation Accuracy During Training Epochs

Now that training has ended, the next step consists in studying how the DenseNet network is learning to classify the input files correctly. To this end, we will plot the average loss, reflecting classification error during validation across epochs and the validation accuracy (Fig. 2).

Script 5. Plotting average loss and validation accuracy

```
### Plot loss and accuracy curves during training

plt.figure("train", (12, 6))
plt.subplot(1, 2, 1)
plt.title("Epoch Average Loss")
x = [i + 1 for i in range(len(epoch_loss_values))]
y = epoch_loss_values
plt.xlabel("epoch")
plt.plot(x, y)
plt.subplot(1, 2, 2)
plt.title("Val Accuracy")
x = [val_interval * (i + 1) for i in range(len(metric_values))]
y = metric_values
plt.xlabel("epoch")
plt.plot(x, y)
plt.show()
```

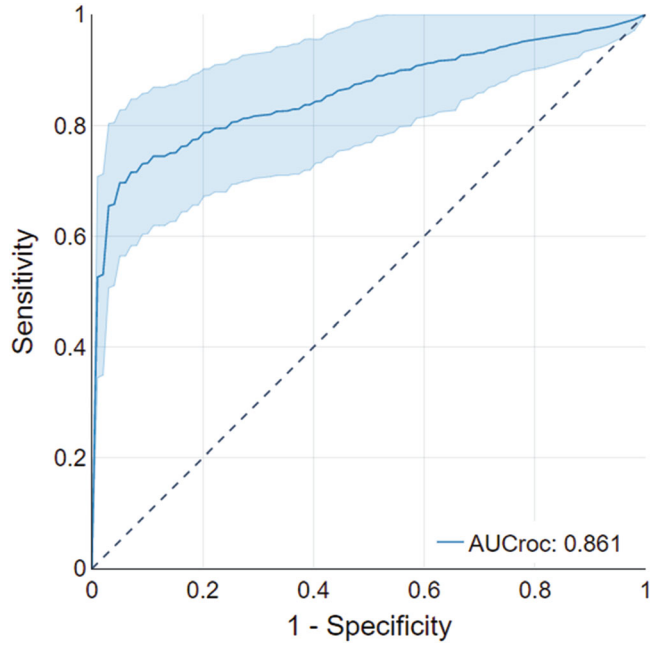

Fig. 3 ROC curve indicating the classification performance of male vs. female discrimination in the testing datasets at various thresholds of sensitivity (true positive rate) and 1-specificity (false positive rate)

3.6 Plot the Test-Set ROC Curve and Get the Area Under the Curve

To have a broader insight regarding the network generalization on a testing dataset, we will plot the receiver operating characteristic (ROC) curve (Fig. 3), representing the diagnostic ability when a discrimination threshold is varied.

Script 6. ROC curve plot

```
### Plot ROC

auc_metric = ROCAUCMetric()
y_pred_trans = Compose([EnsureType(), Activations(softmax=True)])
num_class = 2
y_trans = Compose([EnsureType(), AsDiscrete(to_onehot=num_class)])


for i in range(2):
  test_ds = ImageDataset(image_files = imtest, labels = labtest, transform=val_transforms)
  test_loader = DataLoader(test_ds, batch_size=1, num_workers=2,
pin_memory=torch.cuda.is_available())

  with torch.no_grad():
      y_pred = torch.tensor([], dtype=torch.float32, device=device)
      y = torch.tensor([], dtype=torch.long, device=device)
      for test_data in test_loader:
         test_images, test_labels = (
            test_data[0].to(device),
            test_data[0].to(device),
         )
```

```
            y_pred = torch.cat([y_pred, model(test_images)], dim=0)
            y = torch.cat([y, test_labels], dim=0)
        y_onehot = [y_trans(i) for i in decollate_batch(y)]
        y_pred_act = [y_pred_trans(i) for i in decollate_batch(y_pred)]
        auc_metric(y_pred_act, y_onehot)

numps = Compose([EnsureType(),ToNumpy()])
preds = numps(y_pred_act)
yver = numps(y_true)

a = np.array(preds)
pred = a[:,1]

figure(figsize=(3, 2), dpi=300)
fpr, tpr, threshold = metrics.roc_curve(y_true, pred)
roc_auc = metrics.auc(fpr, tpr)

plt.title('ROC')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

3.7 Plot the Classification Report in a Radar Chart

Now we will get detailed classification report with multiple metrics (Fig. 4): accuracy, sensitivity, specificity, precision, recall, and F1. The last three are particularly important to deal with imbalanced datasets between classes as they provide unbiased performance results.

Script 7. Classification report

```
### Get performance report

y_true = []
y_pred = []

with torch.no_grad():
    for test_data in test_loader:
        test_images, test_labels = (
            test_data[0].to(device),
            test_data[1].to(device),
        )
        pred = model(test_images).argmax(dim=1)
        for i in range(len(pred)):
            y_true.append(test_labels[i].item())
            y_pred.append(pred[i].item())

class_names = ['Female', 'Male']

rep = classification_report(
    y_true, y_pred, target_names=class_names, digits=4, output_dict=True)

### Get radar chart

tp, fn, fp, tn = confusion_matrix(y_true, y_pred).ravel()
acc = (tp+tn)/(tp+tn+fp+fn)
sen = (tp)/(tp+fn)
sp = (tn)/(tn+fp)
```

```
markers = [0, 0.2, 0.4, 0.6, 0.8, 1.0]
proportions = [acc, sp, rep['weighted avg'].get('recall'), rep['weighted avg'].get('f1-score'),
rep['weighted avg'].get('precision'), sen]
labels = ['Accuracy', 'Specificity', 'Recall', 'F1', 'Precision', 'Sensitivity']

def make_radar_chart(name, stats, attribute_labels=labels,
                     plot_markers=markers):

    labels = np.array(attribute_labels)

    angles = np.linspace(0, 2*np.pi, len(labels), endpoint=False)
    stats = np.concatenate((stats,[stats[0]]))
    angles = np.concatenate((angles,[angles[0]]))
    fig = plt.figure()
    ax = fig.add_subplot(111, polar=True)
    ax.plot(angles, stats, 'o-', linewidth=2)
    ax.fill(angles, stats, alpha=0.25)
    ax.set_thetagrids(angles * 180/np.pi, labels, fontsize = 18)
    plt.yticks(markers, fontsize = 18)
    ax.set_title(name, fontsize = 18)
    ax.grid(True)
    fig.set_size_inches(10,10, forward = False)

    return plt.show()

make_radar_chart("Model Performance", proportions)
```

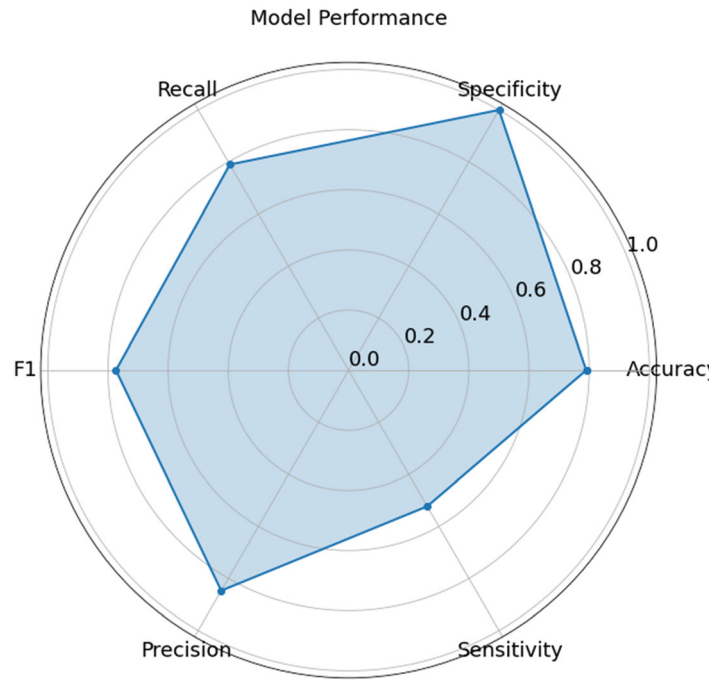

Fig. 4 The radar chart depicts the classification reports, providing a detailed profile of classification performance including accuracy, sensitivity, specificity, precision, recall and F1

3.8 Obtain the Occlusion Sensitivity Map

Finally, we are able to gain insights on what features the DenseNet has been utilizing when classifying subjects. This step is particularly important because we need to be sure that the classification is based on brain anatomy, and not other possible confounding factors such as skull bone structure. Furthermore, information about specific anatomical differences would provide insights regarding biological underpinnings of sex. To this end, we ran *occlusion sensitivity* analyses [16]. The occlusion sensitivity analysis is a technique in convolutional neural networks that is employed to understand what parts of an image are more relevant for deciding a classification output. During this process, small image areas are perturbed by superimposing an occluding mask composed of a gray square. This mask is moved across the image, and the change in probability score for a given class is measured as a function of mask position. When an important part of the image is occluded, the probability score for the predicted class will fall sharply, producing a map of classification relevance.

Script 8. Occlusion sensitivity output interpretation

```
### Run occlusion sensitivity analysis in the sagittal plane (other planes are available, commented below)

msk = 3
stride = 3

for i in range(10):
  # Get a random image and its corresponding label
  img, label = get_next_im()
  print(label)

  # Get the occlusion sensitivity map
  occ_sens = monai.visualize.OcclusionSensitivity(nn_module=model, mask_size=msk, n_batch=10, stride=stride)
  # Only get a single slice to save time.
  # For the other dimensions (channel, width, height), use
  # -1 to use 0 and img.shape[x]-1 for min and max, respectively
  depth_slice = img.shape[2] // 2

  # Sagital
  occ_sens_b_box = [depth_slice-1, depth_slice, -1, -1, -1, -1]

  occ_result, _ = occ_sens(x=img, b_box=occ_sens_b_box)
  occ_result = occ_result[0, label.argmax().item()][None]

  fig = plt.figure(figsize=(15,15))
  plt.xlim([0, 100])
  plt.ylim([0, 100])
  ax = fig.add_subplot(111)
  im = occ_result

  img2 = plt.imshow(ndimage.rotate(img[0, 0, depth_slice, ...].detach().cpu(),-90), interpolation='nearest', cmap='gray', origin='lower')
  img3 = plt.imshow(ndimage.rotate(np.squeeze(im[0][0].detach().cpu()),-90), interpolation='nearest', cmap='gist_heat', origin='lower', alpha = 0.4)
  fig.colorbar(img3)
  plt.show()
```

4 Discussion on Model Example Interpretation

The example displayed in Fig. 5 shows an occlusion sensitivity cluster located in the orbitofrontal cortex. This area seems to be relevant for sex differences. A recent study in more than 2000 MRI scans [17] showed that females, on average, had relatively greater volume in the orbitofrontal cortex when compared to male subjects. This result has been also replicated using state-of-the-art techniques applied to GMV in a large sample ($n = 2838$) and in two independent cohorts, with a large age range, with all data acquired from the same MRI scanner [18]. Importantly, the orbitofrontal cortex is relevant to sex differences in geriatric depression [19], tau deposition in heterozygotes [20], leftward functional connectivity asymmetry [21], and even microsatellite polymorphisms of steroid hormone receptors [22]. Future studies may investigate if these brain anatomical differences are influenced by different factors, including biological heterogeneity or by social conventions generating plastic brain changes through the lifespan.

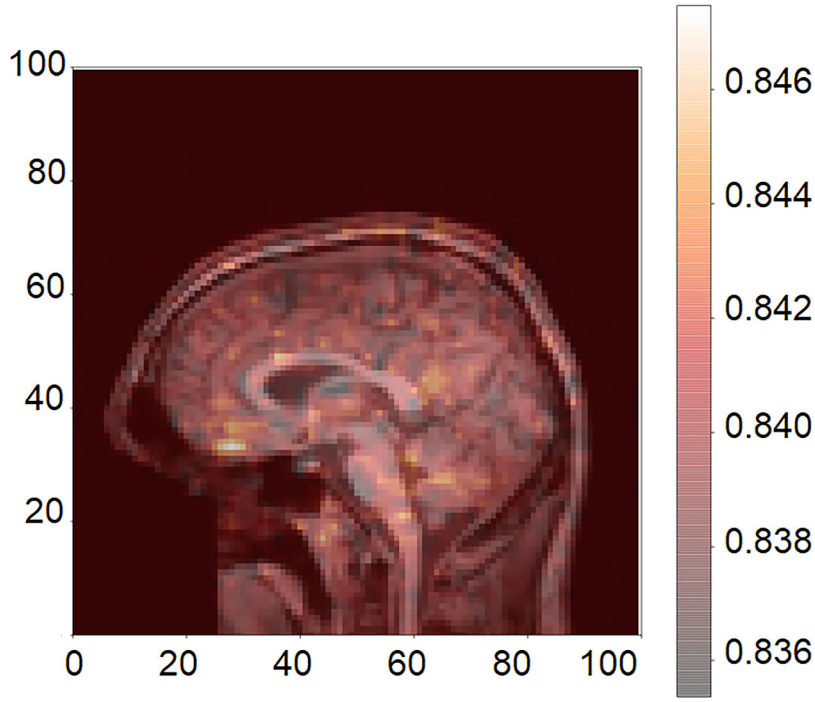

Fig. 5 Sagittal plane T1-weighted MRI image overlaid with the output of the occlusion sensitivity map

5 Conclusions

In this chapter, we described a step-by-step implementation example of a deep learning pipeline based on raw data. After increasing the sample using augmentation, we trained the DenseNet to accurately classify male and female subjects in the test set. In addition, via the occlusion sensitivity analysis, anatomical insights were gained, plus evidence that the classifier was using brain tissue sources and not possible confounding factors. We hope this framework will guide future implementations of different protocols that use neuroimaging data for classification.

References

1. Ching T, Himmelstein DS, Beaulieu-Jones BK, Kalinin AA, Do BT, Way GP, Ferrero E, Agapow P-M, Zietz M, Hoffman MM, Xie W, Rosen GL, Lengerich BJ, Israeli J, Lanchantin J, Woloszynek S, Carpenter AE, Shrikumar A, Xu J, Cofer EM, Lavender CA, Turaga SC, Alexandari AM, Lu Z, Harris DJ, DeCaprio D, Qi Y, Kundaje A, Peng Y, Wiley LK, Segler MHS, Boca SM, Swamidass SJ, Huang A, Gitter A, Greene CS (2018) Opportunities and obstacles for deep learning in biology and medicine. J R Soc Interface 15: 20170387. https://doi.org/10.1098/rsif.2017.0387
2. Pedersen M, Verspoor K, Jenkinson M, Law M, Abbott DF, Jackson GD (2020) Artificial intelligence for clinical decision support in neurology. Brain Commun 2:fcaa096. https://doi.org/10.1093/braincomms/fcaa096
3. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. arXiv:1512.03385. https://doi.org/10.48550/arXiv.1512.03385
4. Shen D, Wu G, Suk H-I (2017) Deep learning in medical image analysis. Annu Rev Biomed Eng 19:221–248. https://doi.org/10.1146/annurev-bioeng-071516-044442
5. Ahmed MR, Zhang Y, Feng Z, Lo B, Inan OT, Liao H (2019) Neuroimaging and machine learning for dementia diagnosis: recent advancements and future prospects. IEEE Rev Biomed Eng 12:19–33. https://doi.org/10.1109/RBME.2018.2886237
6. Huys QJM, Maia TV, Frank MJ (2016) Computational psychiatry as a bridge from neuroscience to clinical applications. Nat Neurosci 19:404–413. https://doi.org/10.1038/nn.4238
7. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 2261–2269
8. Baez S, Flichtentrei D, Prats M, Mastandueno R, García AM, Cetkovich M, Ibáñez A (2017) Men, women. . .who cares? A population-based study on sex differences and gender roles in empathy and moral cognition. PLoS One 12:e0179336. https://doi.org/10.1371/journal.pone.0179336
9. Rutter M, Caspi A, Moffitt TE (2003) Using sex differences in psychopathology to study causal mechanisms: unifying issues and research strategies. J Child Psychol Psychiatry 44:1092–1115. https://doi.org/10.1111/1469-7610.00194
10. Mazure CM, Swendsen J (2016) Sex differences in Alzheimer's disease and other dementias. Lancet Neurol 15:451–452. https://doi.org/10.1016/S1474-4422(16)00067-3
11. Beery AK, Zucker I (2011) Sex bias in neuroscience and biomedical research. Neurosci Biobehav Rev 35:565–572. https://doi.org/10.1016/j.neubiorev.2010.07.002
12. Cahill L (2006) Why sex matters for neuroscience. Nat Rev Neurosci 7:477–484. https://doi.org/10.1038/nrn1909
13. Ruigrok ANV, Salimi-Khorshidi G, Lai M-C, Baron-Cohen S, Lombardo MV, Tait RJ, Suckling J (2014) A meta-analysis of sex differences in human brain structure. Neurosci Biobehav Rev 39:34–50. https://doi.org/10.1016/j.neubiorev.2013.12.004
14. Xiao B, Yang Z, Qiu X, Xiao J, Wang G, Zeng W, Li W, Nian Y, Chen W (2022) PAM-DenseNet: a deep convolutional neural network for computer-aided COVID-19 diagnosis. IEEE Trans Cybern 52:12163–12174. https://doi.org/10.1109/TCYB.2020.3042837

15. Kandel I, Castelli M, Popovič A (2020) Comparative study of first order optimizers for image classification using convolutional neural networks on histopathology images. J Imaging 6:92. https://doi.org/10.3390/jimaging6090092
16. Dyrba M, Hanzig M, Altenstein S, Bader S, Ballarini T, Brosseron F, Buerger K, Cantré D, Dechent P, Dobisch L, Düzel E, Ewers M, Fliessbach K, Glanz W, Haynes J-D, Heneka MT, Janowitz D, Keles DB, Kilimann I, Laske C, Maier F, Metzger CD, Munk MH, Perneczky R, Peters O, Preis L, Priller J, Rauchmann B, Roy N, Scheffler K, Schneider A, Schott BH, Spottke A, Spruth EJ, Weber M-A, Ertl-Wagner B, Wagner M, Wiltfang J, Jessen F, Teipel SJ, ADNI, AIBL, DELCODE study groups (2021) Improving 3D convolutional neural network comprehensibility via interactive visualization of relevance maps: evaluation in Alzheimer's disease. Alzheimers Res Ther 13:191. https://doi.org/10.1186/s13195-021-00924-2
17. Liu S, Seidlitz J, Blumenthal JD, Clasen LS, Raznahan A (2020) Integrative structural, functional, and transcriptomic analyses of sex-biased brain organization in humans. Proc Natl Acad Sci USA 117:18788–18798. https://doi.org/10.1073/pnas.1919091117
18. Lotze M, Domin M, Gerlach FH, Gaser C, Lueders E, Schmidt CO, Neumann N (2019) Novel findings from 2,838 adult brains on sex differences in gray matter brain volume. Sci Rep 9:1671. https://doi.org/10.1038/s41598-018-38239-2
19. Lavretsky H, Kurbanyan K, Ballmaier M, Mintz J, Toga A, Kumar A (2004) Sex differences in brain structure in geriatric depression. Am J Geriatr Psychiatry 12:653–657. https://doi.org/10.1176/appi.ajgp.12.6.653
20. Yan S, Zheng C, Paranjpe MD, Li Y, Li W, Wang X, Benzinger TLS, Lu J, Zhou Y (2021) Sex modifies APOE ε4 dose effect on brain tau deposition in cognitively impaired individuals. Brain 144:3201–3211. https://doi.org/10.1093/brain/awab160
21. Liang X, Zhao C, Jin X, Jiang Y, Yang L, Chen Y, Gong G (2021) Sex-related human brain asymmetry in hemispheric functional gradients. NeuroImage 229:117761. https://doi.org/10.1016/j.neuroimage.2021.117761
22. Tan GC-Y, Chu C, Lee YT, Tan CCK, Ashburner J, Wood NW, Frackowiak RS (2020) The influence of microsatellite polymorphisms in sex steroid receptor genes ESR1, ESR2 and AR on sex differences in brain structure. NeuroImage 221:117087. https://doi.org/10.1016/j.neuroimage.2020.117087