**Mammography Image Classification Using Random Forest and Visual Transformer (ViT)**

**Model Guide for RandomForestViT.py**

1. **Starting with Importing the required libraries**

“import time

…

import ssl

ssl.\_create\_default\_https\_context = ssl.\_create\_unverified\_context”

1. **ViT model definition**

“vit\_model = models.vit\_b\_16(weights=models.ViT\_B\_16\_Weights.DEFAULT)”

* In this section, we are declaring the Visual Transformer (ViT) model using pretrained weights. We will be using this model to extract features from the images in this script.

1. **Transforming section**

“transform = transforms.Compose([

transforms.Resize((224, 224)),

transforms.Grayscale(num\_output\_channels=3),

transforms.ToTensor(),

transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5]),

])

* In this section, we are resizing the images. Having a consistent image size is efficient in terms of computing power, since we are handling large data. Size of 224x224 pixels is often used in this area as we researched.
* As for “num\_output\_channels=3” parameter, eventhough we are using grayscale images, it is stated that various pretrained models can expect 3 channel input images, since they are used in several different areas, not only medical images. By doing so, we try to avoid some compatibility problems.
* “transforms.Normalize” method is used to ensure the consistency.

1. **Dataset class section**

“class MammographyDataset(Dataset):

def \_init\_(self, data\_dir, data\_transform=None):

…

def \_len\_(self):

…

def transform(self, image\_files):

…

def labels(self):

…”

* In this section, we are listing all the files we have in our dataset. Following that, we return the number of images using the “\_len\_” section. “transform” function is used to load and transform the images, fitting them in arrays. In the last section, we are getting an image, executing the work required and we get the label from its name. Extra information can be found in code comments.

1. **Loading dataset, train and test loads**

“print(f"Calculations started...")

counter\_start = time.time() # Counter starts here for calculating total runtime

X = np.array(dataset.transform(dataset.image\_files))

y = dataset.labels()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)”

* Firstly, we are printing out that our main calculations are started using a print function.
* In the following line, we are transforming the images from our dataset into an array and gathering their labels for them.
* Next, we are using %20 of the data for testing. As fort he “random\_state” section, we decided to use a fixed number to ensure that we get healthy results comparing the outcomes from different tests we run, not because we had a different split of test and train.

1. **Gathering features with ViT (train) section**

“vit\_features = []

total\_images = len(X\_train)

for i, img in enumerate(X\_train):

img = torch.tensor(img).unsqueeze(0)

features = vit\_model(img)

vit\_features.append(features.detach().numpy().flatten())

print(f'Extracted features from image {i+1}/{total\_images} ({(i+1)/total\_images\*100:.2f}%)', end='\r')

print()

vit\_features = np.array(vit\_features)”

* We are starting with an empty array to store the features we want, alongside with calculating the total images in training set we created in the previous part.
* Following that, we start a loop to iterate over each image in the training set. After preparing the images for PyTorch, we use the ViT model to gather the features, then storing the features in the list “vit\_features”.
* In the end, we print the progress after each loop.

1. **Training the random forest classifier section**

“rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_model.fit(vit\_features, y\_train)”

* This section is used for the Random Forest model to be trained on the features extracted by ViT in the previous section.
* We are using a fixed “random\_state” value, just like we did in previous classifier models.

1. **Evaluations with ViT (test) section**

“print("Evaluation on the test set:")

vit\_features\_test = []

total\_images\_test = len(X\_test)

for i, img in enumerate(X\_test):

img = torch.tensor(img).unsqueeze(0)

features = vit\_model(img)

vit\_features\_test.append(features.detach().numpy().flatten())

print(f'Extracted features from test image {i+1}/{total\_images\_test} ({(i+1)/total\_images\_test\*100:.2f}%)', end='\r')

print()

vit\_features\_test = np.array(vit\_features\_test)”

* This section is similar to the 6th section. We are using the ViT model to extract features, printing the progress, and store the features in arrays. In this case, we do this process for test images.

1. **Predictions and final section**

“y\_pred\_rf = rf\_model.predict(vit\_features\_test)

accuracy\_rf = accuracy\_score(y\_test, y\_pred\_rf) \* 100

print(f'Accuracy on test set (Random Forest): {accuracy\_rf:.2f}%')

print("Total runtime:", time.time() - counter\_start)”

* In this final section, after gathering features from the previous section, our trained model makes predictions and print the accuracy of that prediction. Alongside with them, we print the total runtime of our script.