



ZZZ Code AI

[Edit Prompt](#)[New Prompt](#)

This website helped you? **Help us back** by sharing your answer and make it viral!

[Copy url](#)

<https://zzzcode.ai/code-explain?id=59a709a7-0283-418b-adff-ebda00d8d3fa>

EF Extensions - EF Core Bulk Methods

[Bulk Insert](#)[Bulk Delete](#)[Bulk Update](#)[Bulk Merge](#)

Do you know this website cost us over **150\$ per day** to stay free?

[Contributing](#) to this website **take 10 seconds**. Simply share one of our sponsor page:

Step 1: Click on [EF Core Add Migration](#)

Step 2: And share the page everywhere!

Medical Image Classification using VGG16

CNN Architecture

Introduction

In this project, we will explore the task of medical image classification using the VGG16 CNN architecture. We will use a dataset of retinal images and train a model to classify them into different categories such as normal, choroidal neovascularization (CNV), diabetic macular edema (DME), and drusen. The VGG16 architecture is a popular choice for image classification tasks due to its deep layers and high performance.

Key Concepts

1. **VGG16 CNN Architecture:** VGG16 is a convolutional neural network architecture that was developed by the Visual Geometry Group at the University of Oxford. It consists of 16 layers, including convolutional layers, max pooling layers, and fully connected layers. VGG16 is known for its simplicity and effectiveness in image classification tasks.
2. **Data Pre-Processing:** Before training the model, we need to pre-process the data. This includes resizing the images to a specific width and height, converting them to RGB format, and normalizing the pixel values. We also use data augmentation techniques such as zooming, flipping, and rotation to increase the diversity of the training data.
3. **Keras Data Generators:** Keras provides a convenient way to load and preprocess image data using data generators. Data generators allow us to load images in batches, apply data augmentation, and perform on-the-fly preprocessing. We can create separate data generators for the training, validation, and testing datasets.
4. **Transfer Learning:** Transfer learning is a technique where we use a pre-trained model as a starting point for our own model. In this project, we use the VGG16 model pre-trained

on the ImageNet dataset. We freeze the pre-trained layers and add our own fully connected layers on top to perform the classification task.

5. **Model Training:** We compile the model with an optimizer, loss function, and evaluation metric. We also define callbacks such as `ModelCheckpoint`, `EarlyStopping`, `CSVLogger`, and `ReduceLROnPlateau` to save the best model, stop training early if the validation loss does not improve, log the training progress, and reduce the learning rate if the validation loss plateaus.
6. **Model Evaluation:** After training the model, we evaluate its performance on the test dataset. We calculate the test loss and accuracy to assess how well the model generalizes to unseen data.

Code Structure

The code provided follows a structured approach to perform medical image classification using the VGG16 CNN architecture. Here is a breakdown of the code structure:

1. **Importing Libraries:** The necessary libraries are imported, including `keras`, `tensorflow`, `pandas`, `numpy`, `matplotlib`, `seaborn`, and `scikit-image`.
2. **Importing Dataset and Data Pre-Processing:** The dataset is imported and the data directories are defined. The number of images in each class is counted and visualized using a bar plot. The tonal distribution of a normal retina image is visualized using a histogram.
3. **Sample Images from the Dataset:** Sample images from each class are displayed to get an idea of the data.
4. **Keras Data Generators:** Data generators are created for the training, validation, and testing datasets. Data augmentation techniques are applied to the training data.

5. Image Data Augmentation: An example of image data augmentation is shown using the ImageDataGenerator.
6. Model: The VGG16 CNN architecture is instantiated and added to a sequential model. The pre-trained layers are frozen and fully connected layers are added on top for classification. The model summary and architecture plot are displayed.
7. Baseline Model Training: The model is compiled with an optimizer, loss function, and evaluation metric. Callbacks are defined for model checkpointing, early stopping, CSV logging, and reducing the learning rate. The model is trained on the training dataset using the TPU strategy.
8. Model Evaluation: The trained model is evaluated on the test dataset to calculate the test loss and accuracy.

Code Examples

Here are a few code examples from the provided code:

1. Creating a Keras Data Generator:

language-python

 Copy code

```
train_datagen = ImageDataGenerator(  
    rescale=1./255,  
    zoom_range=(0.73, 0.9),  
    horizontal_flip=True,  
    rotation_range=10,  
    width_shift_range=0.10,  
    fill_mode='constant',  
    height_shift_range=0.10,
```

```
        brightness_range=(0.55, 0.9),
    )

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    color_mode='rgb',
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True,
    seed=1337
)
```

2. Training the Model:


language-python

 Copy code

```
history = model.fit(
    train_generator,
    epochs=epochs,
    steps_per_epoch=steps_per_epoch,
    validation_data=valid_generator,
    validation_steps=validation_steps,
    verbose=2,
    callbacks=callbacks,
    shuffle=True
)
```

3. Evaluating the Model:

language-python

 Copy code

```
(eval_loss, eval_accuracy) = model.evaluate(test_generator, batch_size=batch_size, verbose=1)
print('Test Loss: ', eval_loss)
print('Test Accuracy: ', eval_accuracy)
```

Conclusion

In this project, we explored the task of medical image classification using the VGG16 CNN architecture. We pre-processed the data, created Keras data generators, performed data augmentation, and trained the model using transfer learning. The model was evaluated on the test dataset to assess its performance. By following this code structure, you can apply the VGG16 architecture to your own medical image classification tasks.

Edit Prompt

New Prompt



284

Share



480

Tweet



28

Share



44

Email



83

Share



134

Print