https://www.kaggle.com/competitions/isic-2024-challenge/overview

The ISIC-2024 Challenge on Kaggle is a skin lesion classification competition. Based on the problem statement, I recommend using a Convolutional Neural Network (CNN) architecture, specifically the EfficientNet model, which has shown excellent performance in image classification tasks, including medical imaging.

Here's a Python code snippet using TensorFlow and Keras to get you started:

import tensorflow as tf

from tensorflow.keras.applications import EfficientNetB7

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define image dimensions and batch size

img\_height, img\_width = 224, 224

batch\_size = 32

# Data augmentation and preprocessing

train\_datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=40,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest'

)

validation\_datagen = ImageDataGenerator(rescale=1./255)

# Load training and validation data

train\_generator = train\_datagen.flow\_from\_directory(

'path/to/train/directory',

target\_size=(img\_height, img\_width),

batch\_size=batch\_size,

class\_mode='categorical'

)

validation\_generator = validation\_datagen.flow\_from\_directory(

'path/to/validation/directory',

target\_size=(img\_height, img\_width),

batch\_size=batch\_size,

class\_mode='categorical'

)

# Build the EfficientNet model

base\_model = EfficientNetB7(weights='imagenet', include\_top=False, input\_shape=(img\_height, img\_width, 3))

x = base\_model.output

x = tf.keras.layers.GlobalAveragePooling2D()(x)

x = tf.keras.layers.Dense(128, activation='relu')(x)

x = tf.keras.layers.Dropout(0.2)(x)

x = tf.keras.layers.Dense(8, activation='softmax')(x) # 8 classes in the competition

model = tf.keras.models.Model(inputs=base\_model.input, outputs=x)

# Compile the model

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Train the model

history = model.fit(

train\_generator,

epochs=10,

validation\_data=validation\_generator

)

This code assumes you have the training and validation data organized in directories, with each class having its own subdirectory. You'll need to adjust the file paths and hyperparameters (e.g., batch size, epochs) according to your specific setup.

Please note that this is just a starting point, and you may need to fine-tune the model and experiment with different architectures or techniques (e.g., transfer learning, data augmentation) to achieve better performance in the competition. Good luck!

Here's an updated code snippet that generates predictions and plots the ROC curve:

import tensorflow as tf

from tensorflow.keras.applications import EfficientNetB7

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import roc\_curve, auc

import matplotlib.pyplot as plt

# ... (rest of the code remains the same as before)

# Make predictions on the validation set

predictions = model.predict(validation\_generator)

# Convert predictions to class probabilities

class\_probabilities = tf.nn.softmax(predictions)

# Get the true labels

true\_labels = validation\_generator.classes

# Compute the ROC curve and AUC for each class

fpr = {}

tpr = {}

roc\_auc = {}

for i in range(8): # 8 classes

fpr[i], tpr[i], \_ = roc\_curve(true\_labels[:, i], class\_probabilities[:, i])

roc\_auc[i] = auc(fpr[i], tpr[i])

# Plot the ROC curve for each class

plt.figure(figsize=(10, 8))

for i in range(8):

plt.plot(fpr[i], tpr[i], label=f'Class {i+1} (AUC = {roc\_auc[i]:.2f})')

plt.plot([0, 1], [0, 1], 'k--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve for Skin Lesion Classification')

plt.legend()

plt.show()

This code generates the ROC curve for each class in the classification problem. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at different thresholds, and the Area Under the Curve (AUC) measures the model's performance.

Please note that you need to have the matplotlib library installed to plot the ROC curve. If you haven't installed it yet, you can do so using pip install matplotlib.

Also, keep in mind that this is just an example code, and you may need to adjust it according to your specific requirements and data.

To download the training images for the ISIC-2024 Challenge, follow these steps:

1. Go to the ISIC-2024 Challenge page on Kaggle: (link unavailable)

2. Click on the "Data" tab.

3. Scroll down to the "Training Data" section.

4. Click on the "Download All" button.

5. Select the "ISIC\_2024\_Training\_Data.zip" file and click "Download".

6. Once the download is complete, extract the zip file to a directory on your local machine.

Alternatively, you can use the Kaggle API to download the data programmatically. Here's an example code snippet:

import kaggle

# Authenticate with Kaggle

kaggle.api.authenticate()

# Download the training data

kaggle.api.dataset\_download\_files('isic-2024-challenge', 'ISIC\_2024\_Training\_Data.zip', path='./data')

Make sure you have the Kaggle API installed (pip install kaggle) and you're authenticated with your Kaggle credentials.

Remember to adjust the path parameter to the desired directory on your local machine.

Once you have the training data downloaded and extracted, you can use the images to train your machine learning model.