Computer Vision in Medicine

An exploration of computer vision methods and their applications in the field of medicine

CAP 6619 - Deep Learning | Project 6 | Summer 2022 - Dr. Marques | Matthew Acs

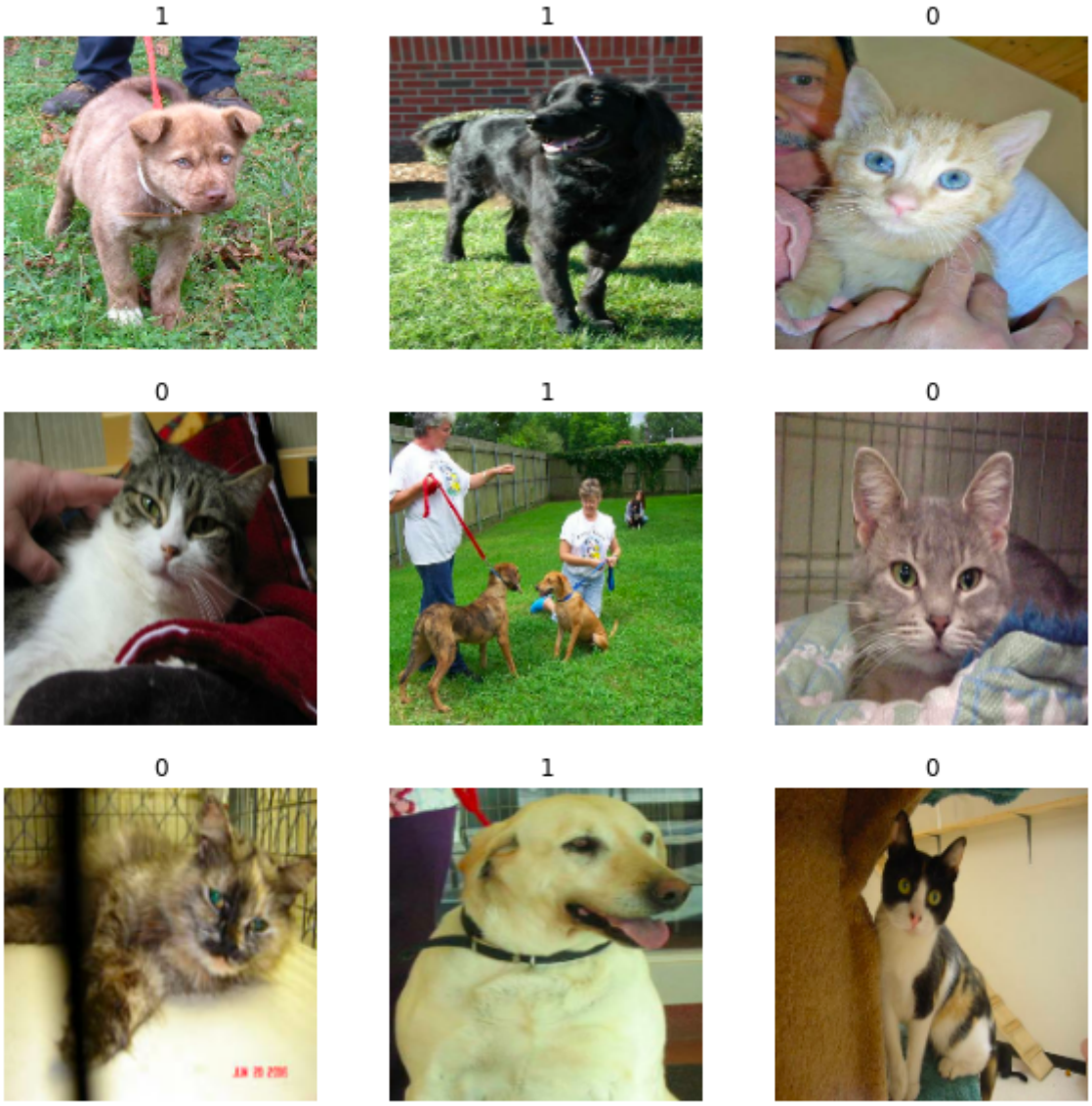
Deep learning techniques for computer vision can be applied to medical imaging to help physicians diagnose pathologies

Introduction

In this assignment I took a deep dive into the recent advancements in computer vision with a focus on their applications in medicine. The four examples that I chose explore how image classification is applied in the field of medicine and what methods are used. The first two examples explore image classification and transfer learning respectivly without touching upon their application in medicine. The third and fourth examples then apply the ideas that were present in the first two examples to medical images. Image classification has the potential to create a significant impact in medicine because it can help physicians identify and diagnose pathologies from medical imaging with greater than human levels of precision. This will help eliminate human error and increase patient care standards. The following examples will provide greater insight into how deep learning is applied to achieve these goals.

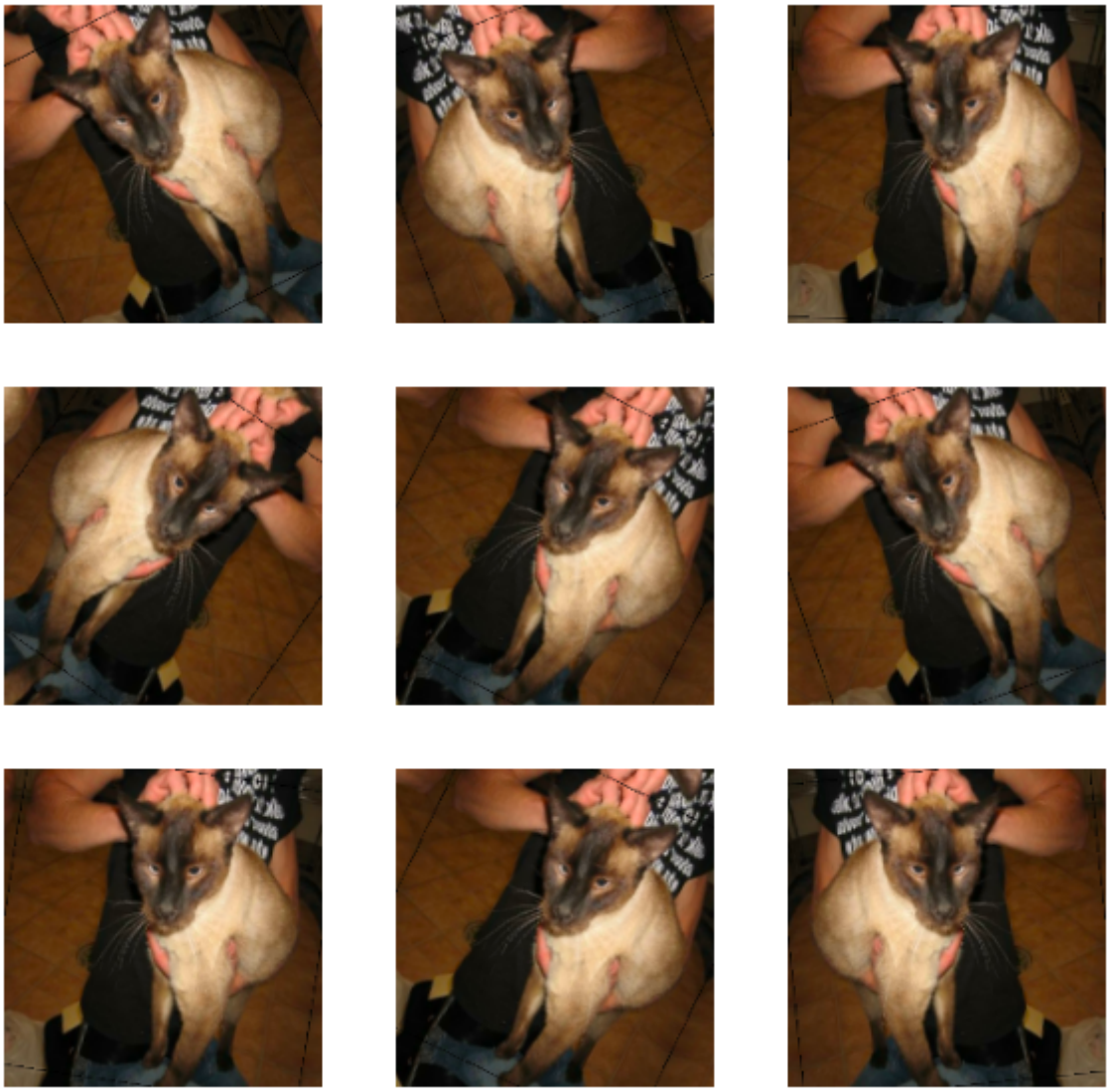
Image Classification and Transfer Learning

The first example demonstrates how an image classifier can be created from scratch using Keras. This classifier differentiates between images of cats and dogs. This example is important because it demonstrates the basics of image classification and utilizes some important techniques. These ideas translate to the creation of an image classifier for medical data as well. For instance, using similar ideas and steps, an image classifier that differentiates between an x-ray of a broken arm and an x-ray of a healthy arm could be created. The main steps that were demonstrated in the example were data preprocessing, model definition and training, and finally model evaluation. The screenshot below shows the data exploration step in the example. This step is important to understand the data so an appropriate model can be created.



One key technique that the example featured is data augmentation. This technique creates more data by distorting a data point to a small degree, artificially increasing the size of the dataset. This promotes generalization because it injects slightly different variations of the already existing data point into the training dataset, encouraging the model to capture the relevant patterns that differentiate between a dog and a cat rather than just memorizing the data points. The code below augments the original images by horizontally flipping the images and applying a random rotation. The result of the data augmentation is shown below. This technique can greatly increase the test accuracy of models in computer vision. When applying this technique to applications in health care, it must be evaluated whether the distorted image still makes sense to feed into the model. For instance, the broken arm x-ray classifier would benefit from data augmentation because the x-ray would still represent a broken arm after a horizontal flip or a slight rotation.

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
    ]
)
```



The first example can be found at:
https://keras.io/examples/vision/image_classification_from_scratch/

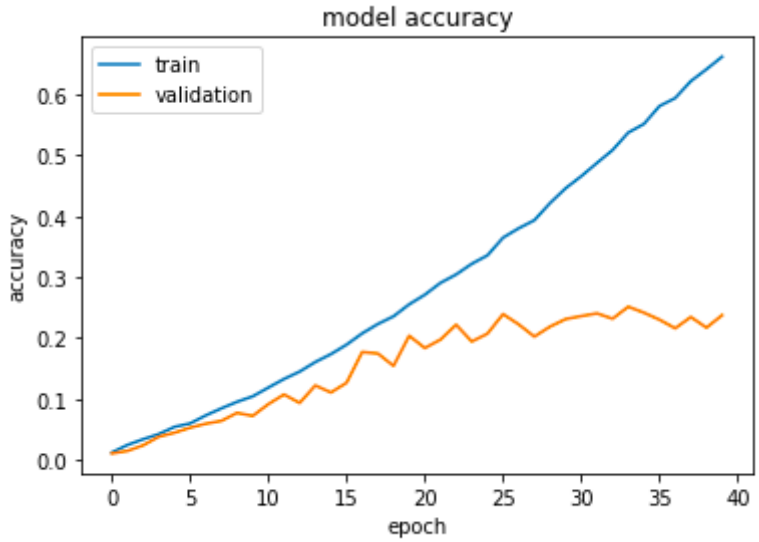
The second example explores how transfer learning can be applied to solve computer vision tasks. Transfer learning is a technique in which a pre-trained model is used to implement a solution to a machine learning problem. This idea is powerful because it can be applied to tasks in which data is sparse while still achieving good results. This can be applied to healthcare to create models more efficiently with higher accuracies. The example uses the pre-trained EfficientNet model using ImageNet weights and fine-tunes the top layers to be specific to the Stanford Dogs dataset. This allows us to use architectures that are known to produce good results for the problem that we face and it allows us to leverage the power of pretrained weights. These pretrained weights at the lower layers are likely to recognize more universal patterns, espcitally if they were trained on a similar task. This allows those layers to be universally useful and only the top layers need to be retrained to

perform well on the current problem. The code below shows how a pre-trained model can be loaded and the top layers can be unfrozen and trianed.

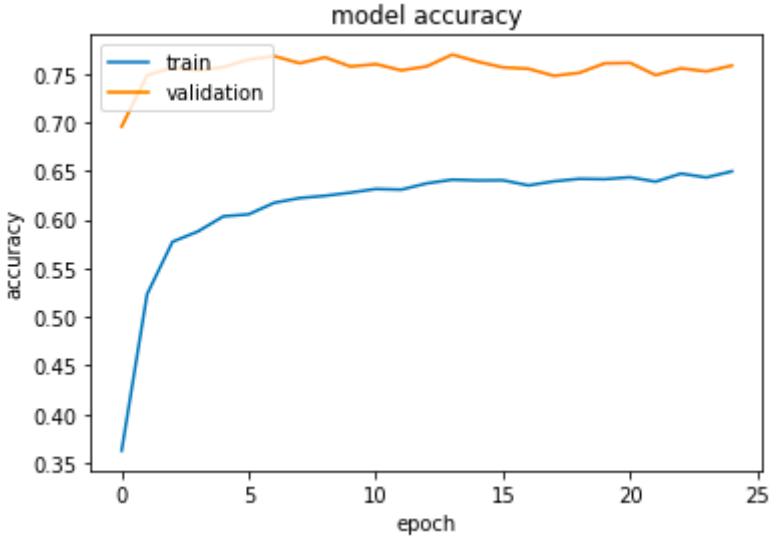
```
def build_model(num_classes):
    inputs = layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
    x = img_augmentation(inputs)
    model = EfficientNetB0(include_top=False, input_tensor=x, weights="imagenet")

    # Freeze the pretrained weights
    model.trainable = False

    # Rebuild top
    x = layers.GlobalAveragePooling2D(name="avg_pool")(model.output)
    x = layers.BatchNormalization()(x)

    top_dropout_rate = 0.2
    x = layers.Dropout(top_dropout_rate, name="top_dropout")(x)
    outputs = layers.Dense(NUM_CLASSES, activation="softmax", name="pred")(x)

    # Compile
    model = tf.keras.Model(inputs, outputs, name="EfficientNet")
    optimizer = tf.keras.optimizers.Adam(learning_rate=1e-2)
    model.compile(
        optimizer=optimizer, loss="categorical_crossentropy", metrics=["accuracy"]
    )
    return model
```

The two graphs below compare the accuracy of a pre-trained model that is fine tuned vs a model that is trained from scratch. The two models have the same architecture, however they have very different accuracies. Clearly, the pre-trained model had a much greater accuracy, showing how transfer learning can be a valuable tool. The techniques behind transfer learning can be applied to many computer vision tasks in healthcare where data may be sparse or for a computationally less expensive solution.

Trained from scratch:



Pre-trained and fine-tuned:



The second example can be found at:
https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/

Pneumonia Classification on a TPU

The third example demonstrates how a CNN can be trained to predict the presence of pneumonia in a chest x-ray. This example creates and trains a model from scratch, much like the first example. The model that was used in this example was a CNN, which is well suited for computer vision tasks. The steps that were used to create the model were essentially the same as for the first

example, which shows that this workflow is similar amongst different applications of computer vision models. The data was initially explored and preprocessed. Then the model was defined, trained, and evaluated. The code below shows the architecture of the model that was used in the example. The model utilizes a combination of convolution and dense blocks that are repeated. This allows the model to extract the necessary patterns from the data.

```
from tensorflow import keras
from tensorflow.keras import layers


def conv_block(filters, inputs):
    x = layers.SeparableConv2D(filters, 3, activation="relu", padding="same")(inputs)
    x = layers.SeparableConv2D(filters, 3, activation="relu", padding="same")(x)
    x = layers.BatchNormalization()(x)
    outputs = layers.MaxPool2D()(x)

    return outputs


def dense_block(units, dropout_rate, inputs):
    x = layers.Dense(units, activation="relu")(inputs)
    x = layers.BatchNormalization()(x)
    outputs = layers.Dropout(dropout_rate)(x)

    return outputs
```

```
def build_model():
    inputs = keras.Input(shape=(IMAGE_SIZE[0], IMAGE_SIZE[1], 3))
    x = layers.Rescaling(1.0 / 255)(inputs)
    x = layers.Conv2D(16, 3, activation="relu", padding="same")(x)
    x = layers.Conv2D(16, 3, activation="relu", padding="same")(x)
    x = layers.MaxPool2D()(x)

    x = conv_block(32, x)
    x = conv_block(64, x)

    x = conv_block(128, x)
    x = layers.Dropout(0.2)(x)

    x = conv_block(256, x)
    x = layers.Dropout(0.2)(x)

    x = layers.Flatten()(x)
    x = dense_block(512, 0.7, x)
    x = dense_block(128, 0.5, x)
    x = dense_block(64, 0.3, x)

    outputs = layers.Dense(1, activation="sigmoid")(x)
```

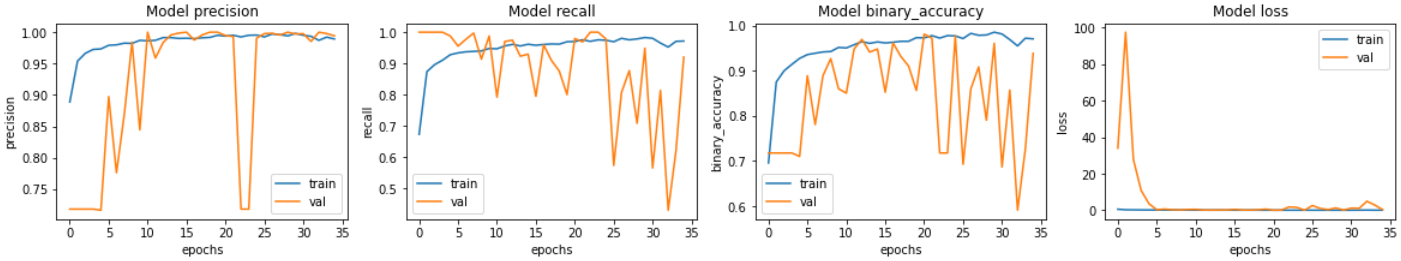
```
    model = keras.Model(inputs=inputs, outputs=outputs)
    return model
```

The example demonstrates a straightforward application of a CNN to a problem within the field of medicine. However, it also highlights two interesting concepts. The first concept is that of a TPU. A TPU or tensor processing unit is a runtime accelerator like a GPU, however, it is designed specifically for machine learning applications. In order to use a TPU on google collab, extra setup is necessary. The setup that the example uses is shown in the code below.

```
import re
import os
import random
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt

try:
    tpu = tf.distribute.cluster_resolver.TPUClusterResolver.connect()
    print("Device:", tpu.master())
    strategy = tf.distribute.TPUStrategy(tpu)
except:
    strategy = tf.distribute.get_strategy()
print("Number of replicas:", strategy.num_replicas_in_sync)
```

```
AUTOTUNE = tf.data.AUTOTUNE
BATCH_SIZE = 25 * strategy.num_replicas_in_sync
IMAGE_SIZE = [180, 180]
CLASS_NAMES = ["NORMAL", "PNEUMONIA"]
```

The second interesting concept that the example highlights is that of overfitting to the validation data. When the model was evaluated on the validation data the accuracy was around 95%. However, the test accuracy was only around 79%, which indicates that the model overfit to the validation data. This is interesting because the validation should be representative of the test data because the model does not train on the validation data. The graphs below show the outcomes of the model training and the output below the graphs shows the test accuracy.



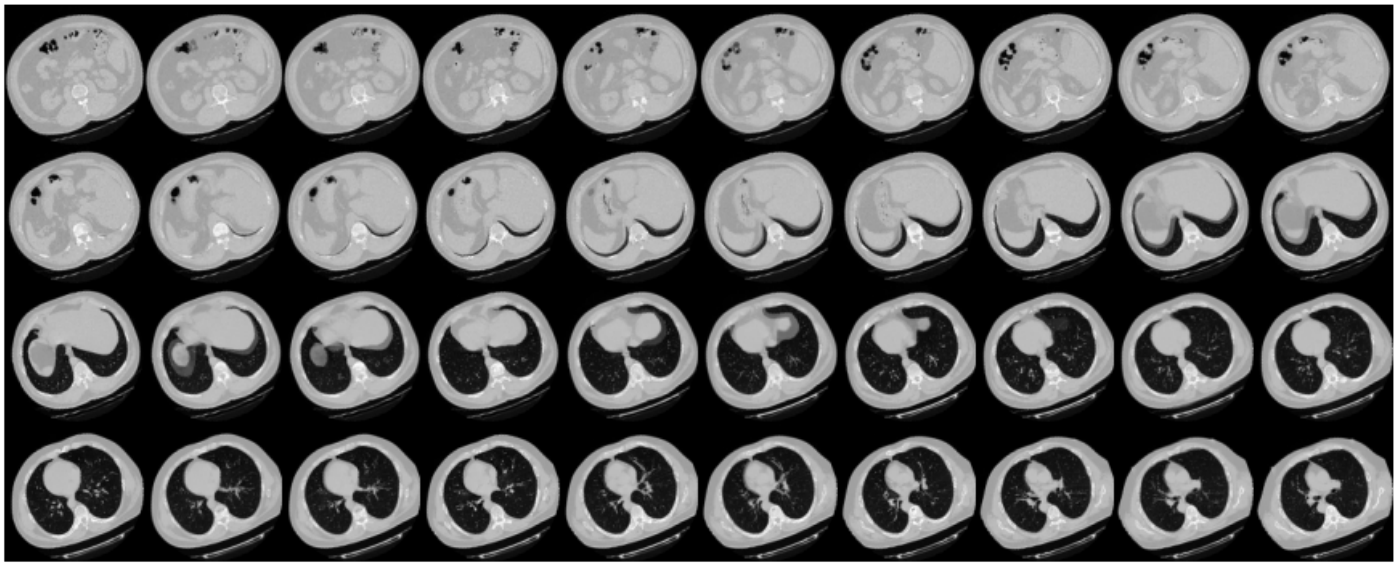
Evaluation of Model on Test Data:

```
4/4 [==============================] - 3s 708ms/step - loss: 0.9718 - binary_accuracy: 0.790

{'binary_accuracy': 0.7900640964508057,
 'loss': 0.9717951416969299,
 'precision': 0.752436637878418,
 'recall': 0.9897436499595642}
```

The third example can be found at: https://keras.io/examples/vision/xray_classification_with_tpus/

3D Image Classification from CT Scans

The final example explores the application of a 3d convolutionary neural network to predict the presence of viral pneumonia associated with COVID-19 in chest CT scans. This example builds upon the previous examples by introducing another dimension to the previous CNNs in the other examples. Furthermore, the task at hand is very similar to the task in the third example, however, it uses 3d CT scans rather than 2d x-rays. This example has elements from each of the previous examples, which shows that many concepts in computer vision are universal across different applications. The model also demonstrates one more way that computer vision can be applied to aid physicians in patient care. This model was created around data from patients with COVID-19, which also shows that computer vision can be a significant tool to help understand new pathologies. COVID-19 created an environment in which research was monumental in helping overcome the challenges of the pandemic and computer vision can provide a rapid way to find novel patterns and gain new insights. The set of images below shows a 3d scan from the dataset. It is essentially a sequence of cross-sectional images that traverse the lungs of the patient.



The architecture of the model must use Cov3d layers rather than 2d layers due to the 3d data that is being used. However, other than the difference in those layers, the architecture follows a similar scheme to other CNNs that we have already seen. The top layers consist of a dense block and the bottom layers are made up of repeating convolutional blocks. Another notable technique that was used in the example was data augmentation. This idea was discussed in the previous examples and was utilized in this example because the dataset was quite small. Data augmentation allows more data points to be considered to help increase the accuracy and generalization power of the model.

```
def get_model(width=128, height=128, depth=64):
    """Build a 3D convolutional neural network model."""

    inputs = keras.Input((width, height, depth, 1))

    x = layers.Conv3D(filters=64, kernel_size=3, activation="relu")(inputs)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)

    x = layers.Conv3D(filters=64, kernel_size=3, activation="relu")(x)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)

    x = layers.Conv3D(filters=128, kernel_size=3, activation="relu")(x)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)

    x = layers.Conv3D(filters=256, kernel_size=3, activation="relu")(x)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)
```

```
    x = layers.GlobalAveragePooling3D()(x)
    x = layers.Dense(units=512, activation="relu")(x)
    x = layers.Dropout(0.3)(x)

    outputs = layers.Dense(units=1, activation="sigmoid")(x)

    # Define the model.
    model = keras.Model(inputs, outputs, name="3dcnn")
    return model


# Build model.
model = get_model(width=128, height=128, depth=64)
model.summary()
```

The fourth example can be found at: https://keras.io/examples/vision/3D_image_classification/

Significance of the Examples

The examples that were explored in this assignment are significant to me because I plan to study the application of artificial intelligence to healthcare and medicine. I am planning on finishing a master's in artificial intelligence before going to medical school, so these examples helped me gain insight into how computer vision is currently being applied to medicine. I am particularly interested in applying computer vision to help doctors in diagnosing pathologies. The power of models to recognize patterns in data and predict if medical images show signs of certain pathologies has the potential to substantially eliminate human error and increase standards of patient care.

Conclusions

Overall, this assignment has shown me that CNNs can be applied to medical images to create models with impressive predictive power. X-rays, CT scans, MRIs, and other medical imaging techniques all have the potential to be used to create models such as the ones that were explored in this assignment. One question that I have after this assignment is how computer vision can be applied to a sequence of medical images throughout time to predict the progression of a pathology. For instance, patients that need knee replacements typically have a form of progressive arthritis. If a sequence of MRIs depicting the state of a patient's knee overtime is obtained, could a model be created to predict how long the patient would have before a knee replacement would be needed?

This type of application would combine imaging with sequential data, which is a common scenario

References

- Photo by Umanoide on Unsplash
- https://keras.io/examples/vision/image_classification_from_scratch/
- https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/
- https://keras.io/examples/vision/xray_classification_with_tpus/
- https://keras.io/examples/vision/3D_image_classification/