ECE 612 - Applied Machine Learning Engineering
Summer 2025
Assignment 3: Transfer Learning for Image Classification

1 Aim of this Assignment

The goal of the assignment is to practice rapidly prototyping and testing transfer learning on the type of vision problems you may one day be asked to solve professionally. The problems have been selected to be of varying difficulty complementary to the examples worked in class.

2 Dataset and Your Objectives

You work for a company that provides software and gear for bird enthusiasts, and have been asked to help with two development projects. The first development project is an integrated camera system users can mount at a position to monitor the feeder. The camera system monitors the presence of birds at the feeder, as well as detects when squirrels or chipmunks are trying to eat from it. The feeders the company builds have doors that can be remotely controlled by the integrated camera system to open or close via WiFi.

Additionally, the company is developing an app to infer types of birds from still images of them. This app will be offered in a stand-alone format for smartphones as well as be integrated into the feeder monitoring camera system to capture bird-type-labelled images of birds at the feeder for review on a smartphone.

Accordingly, your initial assignment working for the company is to build two vision architectures that classify still images:

1. A vision architecture that classifies an image into three c ategories: 1 - containing a bird and no squirrel/chipmunk, 2 - containing a squirrel/chipmuck, or 3 - containing a empty bird-feeder/bird-house/tree.
2. A vision architecture which classifies images of birds among a collection of 358 species of birds.

To help train your architecture you have been given two small datasets of images -

- **Feeder Monitoring Data** - a tfrecords dataset consisting of two collections of images of 3 types (containing squirrels/chipmunks, containing birds without squirrels/chipmunks, containing trees/birdfeeders/birdhouses without birds/squirrels/chipmunks). Please see the template code in §5.1 for how to load the tfrecords files.
- **Bird Classification Data** - a tfrecords dataset consisting of two collections of images of 358 types of birds subsampled from the imagenet-21k dataset and two text labeling files
  - birds-20-eachOf-358.tfrecords: roughly 20 images (a few have less) of each of the 358 types of birds - suggested for use as a training set.
  - birds-10-eachOf-358.tfrecords: 10 images of each of the 358 types of birds - suggested for use as a validation or testing set.
  - birdNames.txt - the names of the types of birds in the order of the 358 classes.
  - birdLabs.txt - the WordNet ids of the bird types in the order of the index of the 358 classes

  Please see the template code in §5.2 for how to load the tfrecords files. By reviewing the images in the tfrecords you will see that the problem is quite difficult as a given type of bird has widely varying appearances, in addition to the fact that birds are animate objects which move substantially and can be captured in photos in many different ways.

The company is aware that the bird classification dataset is woefully small (as internal hand-reviewed datasets often are), and you are at liberty to augment it if you wish so long as you provide your augmented images, although the architecture should be limited to the 358 types of birds identified.

3 Tasks You Must Complete

You will utilize transfer learning to build, train, and save, two deep convolutional neural network models in keras on TensorFlow. The first model will be designed to decide between three outcomes:

- a bird and no squirrel/chipmunk is present in the image

- a squirrel is present in the image
- the image is of a tree/birdfeeder/birdhouse that has no squirrel or bird nearby.

The second model will be designed to take a natural image of a bird of 358 types and return probabilities that it came from each of the 358 types.

4 What You Must Submit

Your submission should be a gzipped or zipped archive titled **abc123-lab2.tgz** or **abc123-lab2.zip** where abc123 is replaced with your Drexel userid. The archive must contain each of the following files

1. preprocessDefinition.py: function that can be applied in a map command on the dataset to make heterogenous images into inputs compatable with your model.
2. buildAndTrainBirdsVsSquirrels.py: loads relevant data, preprocesses it, defines model for (bird-no-squirrel/chipmunk, squirrel/chipmunk, feeder/tree/house-w/o-bird/squirrel/chipmunk) three-way classification, trains the model, saves the model.
3. birdsVsSquirrelsModel.tgz or birdsVsSquirrelsModel.zip a compressed archive of the saved model directory created by the model.save command applied to your model to classify images into the three categories (bird-no-squirrel/chipmunk, squirrel/chipmunk, feeder/tree-w/o-squirrel-or-bird).
4. buildAndTrainBirder.py: loads relevant data, preprocesses it, defines model for 358-way bird-type classification, trains the model, saves the model.
5. birderModel.tgz or birderModel.zip a compressed archive of the saved model directory created by the model.save command applied to your model to classify images of birds into the 358 bird types.
6. if you augment the data you must provide a tfrecords file for your augmented set in the same format as the provided tfrecords file. Your model.fit command must utilize the entire augmented set you provide - do not include images you did not utilize in training or validation.
7. Compare your model's performance with a friend's model and explain why one model is performing better than the other.

5 How Your Assignment Will be Graded

Your grade will consist of the following assessments which will be totaled to create a final score

1. Compliance - was your submission named appropriately and did it have all of the necessary files.
2. Preprocess Code - When placed into a map command applied to the two datasets, does your myPreprocess function produce outputs compatable with your models?
3. Model Training Code assessments: in the code buildAndTrainbirdsVsSquirrelsModel.py and buildAndTrainBirder.py, do you complete at least each of the following
   (a) loads the data, preprocesses it
   (b) load a pre-trained model to transfer from
   (c) modify its top in a manner that includes at the end an appropriate output layer for the task and other hidden layers as you wish.
   (d) freeze the pre-trained weights
   (e) compile the model with an appropriate loss function, optimizer, and metrics
   (f) fit the model
   (g) save the final trained model
4. Birds versus Squirrels Model Performance - How does the accuracy of your model saved in birdsVsSquirrelsModel.tgz or the model trained by your buildAndTrainbirdsVsSquirrelsModel.py compare with the accuracy of that of your peers?
5. Birder Model Performance - How does the average of the top1, top5, top 10, and top 20 accuracies of your model saved in birderModel.tgz or the model trained by your buildAndTrainBirder.py compare with that of your peers? This performance will be measured as described in §5.2 below.
6. Compare your model's performance with a friend's model and explain why one model is performing better than the other.

5.1 Template Code for Loading Birds vs. Squirrels Data

```
import tensorflow as tf

raw_dataset=tf.data.TFRecordDataset(['birds-vs-squirrels-train.tfrecords'])

feature_description={'image':tf.io.FixedLenFeature([],tf.string),
                     'label':tf.io.FixedLenFeature([],tf.int64)}

def parse_examples(serialized_examples):
    examples=tf.io.parse_example(serialized_examples,feature_description)
    targets=examples.pop('label')
    images=tf.image.resize_with_pad(tf.cast(tf.io.decode_jpeg(
        examples['image'],channels=3),tf.float32),299,299)
    return images, targets

#you can edit batch size and num_parallel calls below based on your architecture
dataset=raw_dataset.map(parse_examples,num_parallel_calls=16).batch(128)
```

5.2 Template Code for Birder Performance Measurement

```
import tensorflow as tf
from preprocessDefinition import preprocess

# file below will be replaced by a testing set of identical format
raw_dataset=tf.data.TFRecordDataset(['birds-20-eachOf-358.tfrecords'])

feature_description={'image':tf.io.FixedLenFeature([],tf.string),
                     'birdType':tf.io.FixedLenFeature([],tf.int64)}

def parse_examples(serialized_examples):
    examples=tf.io.parse_example(serialized_examples,feature_description)
    targets=examples.pop('birdType')
    images=tf.image.resize_with_pad(tf.cast(tf.io.decode_jpeg(
        examples['image'],channels=3),tf.float32),299,299)
    return images, targets

#you can edit batch size and num_parallel calls below based on your architecture
dataset=raw_dataset.map(parse_examples,num_parallel_calls=16).batch(128)

# run through your provided preprocess method
dataset=dataset.map(preprocess,num_parallel_calls=16).cache()

#load the model you saved
model=tf.keras.models.load_model('birder')

#ensure the metrics included are the ones we wish to evaluate
top5err=tf.keras.metrics.SparseTopKCategoricalAccuracy(k=5,name='top5')
top10err=tf.keras.metrics.SparseTopKCategoricalAccuracy(k=10,name='top10')
top20err=tf.keras.metrics.SparseTopKCategoricalAccuracy(k=20,name='top20')

model.compile(loss='sparse_categorical_crossentropy',optimizer=opt,
        metrics=['accuracy',top5err,top10err,top20err])

#returns the loss and metrics evaluated on the dataset
resp=model.evaluate(dataset)
```