

Transfer learning and fine-tuning with pre-trained MobileNetV2 network

1. Configure dataset for performance

- [`tf.data.Dataset.prefetch`](#): decouple time when data produced from time when data consumed
 - Prefetch and preprocess next set of inputs as model training on previous set
 - Background thread and internal buffer to prefetch elements from input dataset ahead of request
 - Num of elts to prefetch \geq num of batches consumed by a single training step
 - Manually tune or use `tf.data.AUTOTUNE`: prompts `tf.data` runtime to tune value dynamically at runtime

2. Data augmentation

- [`tf.keras.layers.RandomFlip`](#): randomly flips images during training
 - Arguments
 - `mode`: string indicating which flip mode to use (e.g., "horizontal" for left-right)
 - `seed`
- [`tf.keras.layers.RandomRotation`](#): randomly rotates images during training
 - Arguments
 - `seed`
 - `factor`: float represented as fraction of 2π or tuple of size 2 for lower and upper bound for rotating clockwise and counter-clockwise (if one float then the same for both)
- [`tf.keras.layers.RandomContrast`](#): preprocessing layer that randomly adjusts contrast during training
 - Arguments
 - `factor`: positive float represented as a fraction of value or tuple of size 2 for lower and upper bounds
 - When a single float then randomly pick between $[1.0 - \text{lower}, 1.0 + \text{upper}]$
 - `seed`

3. Rescale pixel values

- [`tf.keras.applications.mobilenet_v2.preprocess_input`](#): returns preprocessed `numpy.array` or `tf.Tensor` with input pixel values scaled between -1 and 1
- Alternative: `tf.keras.layers.Rescaling(1./127.5, offset = 0)`

4. Create base MobileNet model from pre-trained convolution network

- MobileNetV2 model trained on the ImageNet dataset
- Pick layer of MobileNetV2 to use for feature extraction
 - Use last layer before flattening → **bottleneck layer** (more generality vs. final / top layer)
 - Specify `include_top = False` to remove last classification layer(s) on "top"
- `next(iterator[, default]):` return the next item from the iterator

5. Feature extraction

- **5a.** Freeze convolutional base model → set `layer.trainable = False` (or `base_model.trainable = False` for all layers) to prevent weights from being updated during training
- **5b.** Add layers to allow for classification of images
 - Average over 7x7 spatial locations with [tf.keras.layers.GlobalAveragePooling2D](#) (global average pooling for spatial data)
 - Arguments
 - `keepdims=False`: whether to keep spatial dimensions → default `False` ∴ output: 2D tensor with shape `(batch_size, channels) = (32,`
 - Add `tf.keras.layers.Dense` to convert features into a single prediction per image
 - Use activation function "softmax" to return probabilities image is of a particular class for 18 classes (i.e., places)
- **5c.** Build a model by chaining together data augmentation, rescaling, `base_model`, feature layers using the Keras Functional API
 - Note: `training = False` for `base_model` since model contains a `BatchNormalization` layer → runs in inference mode and does not update its mean and variance stats
 - [Keras functional API](#): create more flexible models than `tf.keras.Sequential` API
 - Builds a graph of layers since deep learning model (usually) as directed acyclic graph (DAG) of layers
 - Create input node: `inputs = keras.Input(shape = (784,))`
 - Contains info about shape, dtype of input data
 - Create new node in graph of layers by calling a layer on inputs object: `dense = layers.Dense(64, activation = "relu")` \\\ `x = dense(inputs)`
 - Layer call draws an arrow from inputs to new layer → pass inputs to `dense` layer and get `x` as the output
 - Add another hidden layer: `x = layers.Dense(64, activation = "relu")(x)`

- Pass `x` to next hidden layer and store output as `x`
- Add output layer: `outputs = layers.Dense(10)(x)`
- Create a model: `model = keras.Model(inputs = inputs, outputs = outputs)`
- Training, evaluation, inference the same as `Sequential` models

6. Compile the model

- `model.compile(optimizer, loss, metrics)`
 - optimizer: [tf.keras.optimizers.Adam\(learning_rate = base_learning_rate\)](#) → implements Adam algorithm, a stochastic gradient descent method based on adaptive estimation of first- and second-order moments
 - Higher computational efficiency, lower memory requirements
 - loss: "sparse_categorical_crossentropy" → use for two or more classes when labels are integers
 - [tf.keras.losses.SparseCategoricalCrossentropy](#)
 - metrics: [\['accuracy'\]](#) → (num times preds = labels) / count

7. Train the model

8. Fine tuning

- Only training a few layers on top of `MobileNetV2` base model → can increase performance to train weights of top layers of pre-trained `base_model` to force more generic params to associate with specific dataset
 - Only fine-tune small num of top layers (not whole `MobileNetV2` model) → higher up (later) layers more specialized ∴ more specific to dataset on which model trained
- **8a** Un-freeze the top (i.e., later) layers of the `base_model` (the `MobileNetV2` instance) but keep bottom (i.e., earlier) layers untrainable
- **8b.** Re-compile the model → use a lower learning rate to prevent model from overfitting quickly
 - [tf.keras.optimizers.RMSprop](#): RMSprop algorithm maintains a moving (discounted_ average of square of gradients → divides gradient by root of average
- **8c.** Continue training the model → use argument `initial_epoch` to start training again based on training model to convergence previously
 - `initial_epoch = history.epoch[-1]` → begin at last epoch of first round of training (when all `base_model` params frozen)

Model Checkpoints:

- [tf.keras.callbacks.ModelCheckpoint](#): callback to save keras model or model weights at some frequency
 - Use to save the weights that yield the maximum validation accuracy since model may begin to overfit to the training data \therefore the last epoch may not have the weights that yield the best validation accuracy
 - Access the weights: `model.load_weights(checkpoint_path)`
- [tf.keras.callbacks.EarlyStopping](#): stop training the model when a monitored metric has stopped improving
 - `patience`: number of epochs without improvement to wait until the training process stops

Saving and Loadings Models

- Save just the weights will callbacks (i.e., `tf.keras.callbacks.ModelCheckpoint`) or [save the entire model](#) as a `SavedModel`
 - Make a directory: `mkdir -p saved_model`
 - Save the model using the directory
`path: model.save('saved_model/my_model')`
 - Load a new instance of the model: `new_model = tf.keras.models.load_model('saved_model/my_model')`

Tensorflow on Rivanna

- TensorFlow on Rivanna: <https://www.rc.virginia.edu/userinfo/rivanna/software/tensorflow/>
- UVA software containers (creating singularity containers) and docker hub info: <https://www.rc.virginia.edu/userinfo/rivanna/software/containers/>
- Docker images on Rivanna: <https://www.rc.virginia.edu/userinfo/howtos/rivanna/docker-images-on-rivanna/>
- `uvarc/rivanna-docker`: <https://github.com/uvarc/rivanna-docker>
- Explanation for code `singularity pull docker://uvarc/tensorflow:2.8.0` to create local copy of container image in a local directory: <https://hub.docker.com/r/uvarc/tensorflow>

Additional sources for a different approach (when data is stored with a directory structure s.t. images for each class are in separate folders)

- From the DEEPLIZARD website and YouTube channel
 - [MobileNet Image Classification With TensorFlow's Keras API](#) → default size for `MobileNet`: (224, 224)
 - [Process Images For Fine-Tuned MobileNet With TensorFlow's Keras API](#)

- [Fine-Tuning MobileNet On Custom Data Set With TensorFlow's Keras API](#)

Organizing and Correctly Labeling Files

- Search for matches with regex using `re.search`: <https://stackabuse.com/python-check-if-string-contains-substring/>
- Multiple if else conditions for conditional col: <https://stackoverflow.com/questions/48569166/multiple-if-else-conditions-in-pandas-dataframe-and-derive-multiple-columns>
- Pandas - dataframe groupby - how to get sum of multiple columns (using `agg` and a `lambda` function): <https://stackoverflow.com/questions/46431243/pandas-dataframe-groupby-how-to-get-sum-of-multiple-columns>
- Counting the number of files in a directory with Linux using `ls | wc -l`: <https://devconnected.com/how-to-count-files-in-directory-on-linux/>
- How to convert two columns of a dataframe to a dictionary with `dict(zip(df.col1, df.col2))`: <https://cmdlinetips.com/2021/04/convert-two-column-values-from-pandas-dataframe-to-a-dictionary/>
- Listing files in a directory with `os.listdir('dir_name')`: <https://www.codegrepper.com/code-examples/python/pandas+list+files+in+directory>
- Using the `os` module: <https://docs.python.org/3/library/os.html>
- Copying files in Python with the `shutil` module: <https://stackoverflow.com/questions/123198/how-to-copy-files>
- Using the `shutil` module: <https://docs.python.org/3/library/shutil.html>
- Dictionary comprehensions: <https://www.datacamp.com/community/tutorials/python-dictionary-comprehension>
- Create multiple subdirectories in Linux with one command (`mkdir -p {dir1,dir2,...}`): <https://www.howtogeek.com/275069/how-to-create-multiple-subdirectories-with-one-linux-command/>
- Only listing subdirectories in a directory in Linux with `ls -d */`: <https://www.cyberciti.biz/faq/linux-list-just-directories-or-directory-names/>
- Number rows in a group in increasing order in pandas: <https://stackoverflow.com/questions/37997668/pandas-number-rows-within-group-in-increasing-order>

- List only files in the current directory with `os.listdir()` and `os.path.isfile()`: <https://stackoverflow.com/questions/11968976/list-files-only-in-the-current-directory>
- Download files from Google Colab to Google Drive with `from google.colab import files`: <https://predictivehacks.com/?all-tips=how-to-download-files-and-folders-from-colab>

Handling Imbalanced Data

- Calculating class weights to use when modeling with imbalanced data: https://www.tensorflow.org/tutorials/structured_data/imbalanced_data#train_a_model_with_class_weights
- Creating power labels and function to over/undersample classes to balance the data: <https://medium.com/the-owl/imbalanced-multilabel-image-classification-using-keras-fbd8c60d7a4b>

Multi-label classification websites

- Classification on an imbalanced dataset: https://www.tensorflow.org/tutorials/structured_data/imbalanced_data
- Multi-label classification with NN keras: <https://towardsdatascience.com/multi-label-image-classification-with-neural-network-keras-ddc1ab1afede>
- Multi-label classification in tensorflow: <https://towardsdatascience.com/multi-label-image-classification-in-tensorflow-2-0-7d4cf8a4bc72>
- **Multi-label classification in tensorflow: <https://medium.com/deep-learning-with-keras/how-to-solve-multi-label-classification-problems-in-deep-learning-with-tensorflow-keras-7fb933243595>
- Video version of above article: <https://www.youtube.com/watch?v=5MQ63pDxULw>
- Google Colab notebook: https://colab.research.google.com/drive/1dpojRslAA15q4tswH_9j8WlQRo0BITSp?usp=sharing#scrollTo=Q3iCuaVXNLhS
- tf.data.Dataset API: https://www.tensorflow.org/api_docs/python/tf/data/Dataset
 - `from_tensor_slices`
- tf.io.decode_jpeg: https://www.tensorflow.org/api_docs/python/tf/io/decode_jpeg
- uint8: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-dtyp/a88ed362-a905-4ed2-85f5-cfc8692c9842#:~:text=A%20UINT8%20is%20an%208,is%20not%20reserved%20for%20signing
- tf.io.read_file: https://www.tensorflow.org/api_docs/python/tf/io/read_file
- tf.py_function: https://www.tensorflow.org/api_docs/python/tf/py_function

- How to use the `tf.data.Dataset.map()` function in TensorFlow: <https://www.gcptutorials.com/article/how-to-use-map-function-with-tensorflow-datasets>
- Prefetching: https://www.tensorflow.org/guide/data_performance#prefetching
- `numpy.squeeze`: <https://numpy.org/doc/stable/reference/generated/numpy.squeeze.html>
- Combinations of activation and loss functions in different situations: <https://towardsdatascience.com/deep-learning-which-loss-and-activation-functions-should-i-use-ac02f1c56aa8>

Ensemble Models

- Workflow for normalizing and testing weights for ensemble models: <https://machinelearningmastery.com/weighted-average-ensemble-for-deep-learning-neural-networks/#:~:text=A%20weighted%20ensemble%20is%20an,the%20performanc,e%20of%20the%20model>
- `itertools.product(*iterables, repeat=1)`: <https://docs.python.org/3/library/itertools.html#itertools.product>
- Element-wise multiplication of two lists with `zip`: <https://stackoverflow.com/questions/10271484/how-to-perform-element-wise-multiplication-of-two-lists>
- `tf.keras.layers.Rescaling` to rescale pixel values within neural network: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Rescaling
- Creating an ensemble model from pre-trained models: <https://www.tensorflow.org/guide/keras/functional>
- Renaming a model with `model._name = "NAME"` to prevent errors about models being named the same thing in ensemble model: <https://stackoverflow.com/questions/56886442/error-when-trying-to-rename-a-pretrained-model-on-tf-keras>

GitHub

- Creating a personal access token (instead of using a password): <https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>
- Git Large File Storage: <https://git-lfs.github.com/>
 - **Steps**
 - Download v3.1.4 (mac - Intel Silicon)
 - `brew install git-lfs`
 - `git lfs install`
 - `cd ../Desktop/repos`
 - `git clone https://github.com/cew4pf/DS6050-eye-project.git`
 - `cd DS6050-eye-project/`

- `git lfs track "*.zip"`
- `git add .gitattributes`
- `git add test_dir.zip`
- `git commit -m "Zipped data created with Google Colab"`
- `git push origin main`