**Dataset/Process Notes**

We want to gather up to 1000 images per construction material label by using one (or multiple) of the following methods (in order of desirability):

1. Google building datasets or groups of easily downloadable building images
2. Downloading/using county/tax/state databases for building images and construction materials
3. Downloading building images in bulk from Google images using code
4. Asking state/county officials for their images on buildings along with construction types, or even the whole dataset they have
5. Going through and getting/annotating images manually
6. Webscraping tax databases and other sites (most difficult, don't want to do)

Afterword, I intend to use transfer learning, and have the above collected images used to specialize the model to our classification task.

Starting with classification of Construction Materials:  
1. Brick   2. Wood   3. Glass   4. Concrete   5. Steel   6. Stone   (Also had 7. None and 8. Delete)

Annotation Set:

* 110 images from each of 84ish categories of building types (10 extra images to replace any that are deleted)
* Each user given a set of 110-220 images to annotate in a sitting

-Will first ask user about about 2 well-known images to test their quality of annotation  
-Then, user will be asked what majority/primary material is there  
-Then after that, user will be asked what second majority/secondary material is found, or if there is none (or not enough to be significant)  
-User will also have the choice to delete an image, if it is deemed to be not useful based on some criteria  
-WE WILL ALSO BE CLASSIFYING BUILDING USE AT THE SAME TIME (SO program will ask about building use right after question for construction material)

**Model Development Notes**

**Settings**

1. Paths start here: **C:/Users/mkarimi/OneDrive - Environmental Protection Agency (EPA)/ImageAnnotation/**
2. Image dimensions  = 256x256 (most images are at this or above this, very few below)

**Preprocessing**

1. Need to determine how to feed labels to system
   1. If use flow\_from\_directory, the directory names are used as the labels, so I have developed a script that will do that
      1. Have also placed folders in the ImageAnnotation folder which correspond to this method

**Training**

1. For now I am including images marked "Deleted"
   1. Update: No longer including these
2. Training on Mr. Boe's Desktop at RTP

**Model Performance Improvement Notes/Options:**

* hyperparameter tuning
  + batchsize
  + epochs
  + horizontalflip
  + rotationrange
  + zoomrange
  + widthshiftrange
  + heightshiftrange
  + model layers (and its #'s)
  + optimizer
  + stepsperepoch
  + validationsteps
  + loss function
* img size
* dropout
* augmentation (already mentioned above)
* callbacks: model checkpoints, early stopping, learning rate scheduler
* transfer learning
* progressive resizing

**Results**

* Look at this file: [Results](https://teams.microsoft.com/l/file/F460763D-0BCE-4B96-B25A-4197D1F50A41?tenantId=88b378b3-6748-4867-acf9-76aacbeca6a7&fileType=xlsx&objectUrl=https%3A%2F%2Fusepa.sharepoint.com%2Fsites%2FTaha%2FShared%20Documents%2FGeneral%2FModelTrainingNotes.xlsx&baseUrl=https%3A%2F%2Fusepa.sharepoint.com%2Fsites%2FTaha&serviceName=teams&threadId=19:87b1eb5d8084441691064c154bafcd85@thread.skype&groupId=05be4e14-6942-44c3-b46e-a66b5165f7c6)

# Image Augmentation Options

More Information: <https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator>

Example:

tf.keras.preprocessing.image.ImageDataGenerator(

    featurewise\_center=False, samplewise\_center=False,

    featurewise\_std\_normalization=False, samplewise\_std\_normalization=False,

    zca\_whitening=False, zca\_epsilon=1e-06, rotation\_range=0, width\_shift\_range=0.0,

    height\_shift\_range=0.0, brightness\_range=None, shear\_range=0.0, zoom\_range=0.0,

    channel\_shift\_range=0.0, fill\_mode='nearest', cval=0.0, horizontal\_flip=False,

    vertical\_flip=False, rescale=None, preprocessing\_function=None,

    data\_format=None, validation\_split=0.0, dtype=None

)

Augments:

|  |  |
| --- | --- |
| featurewise\_center | Boolean. Set input mean to 0 over the dataset, feature-wise. |
| samplewise\_center | Boolean. Set each sample mean to 0. |
| featurewise\_std\_normalization | Boolean. Divide inputs by std of the dataset, feature-wise. |
| samplewise\_std\_normalization | Boolean. Divide each input by its std. |
| zca\_epsilon | epsilon for ZCA whitening. Default is 1e-6. |
| zca\_whitening | Boolean. Apply ZCA whitening. |
| rotation\_range | Int. Degree range for random rotations. |
| width\_shift\_range | Float, 1-D array-like or int   * float: fraction of total width, if < 1, or pixels if >= 1. * 1-D array-like: random elements from the array. * int: integer number of pixels from interval (-width\_shift\_range, +width\_shift\_range) * With width\_shift\_range=2 possible values are integers [-1, 0, +1], same as with width\_shift\_range=[-1, 0, +1], while with width\_shift\_range=1.0 possible values are floats in the interval [-1.0, +1.0). |
| height\_shift\_range | Float, 1-D array-like or int   * float: fraction of total height, if < 1, or pixels if >= 1. * 1-D array-like: random elements from the array. * int: integer number of pixels from interval (-height\_shift\_range, +height\_shift\_range) * With height\_shift\_range=2 possible values are integers [-1, 0, +1], same as with height\_shift\_range=[-1, 0, +1], while with height\_shift\_range=1.0 possible values are floats in the interval [-1.0, +1.0). |
| brightness\_range | Tuple or list of two floats. Range for picking a brightness shift value from. |
| shear\_range | Float. Shear Intensity (Shear angle in counter-clockwise direction in degrees) |
| zoom\_range | Float or [lower, upper]. Range for random zoom. If a float, [lower, upper] = [1-zoom\_range, 1+zoom\_range]. |
| channel\_shift\_range | Float. Range for random channel shifts. |
| fill\_mode | One of {​​​​​"constant", "nearest", "reflect" or "wrap"}​​​​​. Default is 'nearest'. Points outside the boundaries of the input are filled according to the given mode:   * 'constant': kkkkkkkk|abcd|kkkkkkkk (cval=k) * 'nearest': aaaaaaaa|abcd|dddddddd * 'reflect': abcddcba|abcd|dcbaabcd * 'wrap': abcdabcd|abcd|abcdabcd |
| cval | Float or Int. Value used for points outside the boundaries when fill\_mode = "constant". |
| horizontal\_flip | Boolean. Randomly flip inputs horizontally. |
| vertical\_flip | Boolean. Randomly flip inputs vertically. |
| rescale | rescaling factor. Defaults to None. If None or 0, no rescaling is applied, otherwise we multiply the data by the value provided (after applying all other transformations). |
| preprocessing\_function | function that will be applied on each input. The function will run after the image is resized and augmented. The function should take one argument: one image (Numpy tensor with rank 3), and should output a Numpy tensor with the same shape. |
| data\_format | Image data format, either "channels\_first" or "channels\_last". "channels\_last" mode means that the images should have shape (samples, height, width, channels), "channels\_first" mode means that the images should have shape (samples, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last". |
| validation\_split | Float. Fraction of images reserved for validation (strictly between 0 and 1). |
| dtype | Dtype to use for the generated arrays. |

# Keras Tuner

If I were to try to use the Keras Tuner, here are some guides to it:

* <https://medium.com/analytics-vidhya/keras-tuner-hyperparameter-tuning-made-easy-50cf050ac6e3>
* <https://www.sicara.ai/blog/hyperparameter-tuning-keras-tuner>

***3rd Party AI Options***

**H20.ai's Driverless AI**

* looks very well built, high quality, plenty of info provided and great UI
  + no clear cost information, but some quotes related to Amazon AWS reservations for Driverless put it around $14/hr or around $60,000/yr
  + apparently can do image classification
    - we can also test it with our own data for free
  + Need to ask about cost

**AutoAI by Watson Studio**

* looks potentially useful as well, also looks well built and provides information during the training process
  + Two estimates seen for cost:
    - $500/month/user
    - Standard - $99/month for 50capacity unit hrs per month and then pay as you go after that
    - Enterprise - $6000/month for 5000capacity unit hrs per month and then pay as you go after that
    - Both of above have full access to Watson Studio Tools
  + Not sure if it can do image classification
  + Has a 'demo/try me' option, but wasn't able to get it to work

**AWS SageMaker Studio Autopilot**

* looks good, seems like it should be able to do multiclass image classification (although not sure how I would input the data)
* seems to have good UI
* It has ability to fine-tune a trained model (like ResNet) with our own dataset, so basically transfer learning, and thus may also have the ability to do multiclass image classification, so a decent option I think
* seems to have an easy way to deploy data
* Not sure about the cost though - need to find out

**Google's autoML**

* Vision is an image classification tool that it has
* Can run a free trial on it and deploy it