Cool, now that you have a baseline for Cifar10, let's get started with feature extraction.

The Code

```
import pickle
import tensorflow as tf
# TODO: import Keras layers you need here
```

Nothing fancy here, just some imports we need to run the code, `pickle` is used to load the bottleneck features.

```
flags = tf.app.flags
FLAGS = flags.FLAGS

# command line flags
flags.DEFINE_string('training_file', '', "Bottleneck features trainin
g file (.p)")
flags.DEFINE_string('validation_file', '', "Bottleneck features valid
ation file (.p)")
```

Here we define some *command line flags*, this avoids having to manually open and edit the file if we want to change the files we train and validate our model with.

Here's how you would run the file from the command line:

```
python feature_extraction.py --training_file vgg_cifar10_100_bottlene
ck_features_train.p --validation_file vgg_cifar10_bottleneck_features
_validation.p
```

Running this program will train feature extraction with the VGG network/Cifar10 dataset bottleneck features. The 100 in `vgg_cifar10_100` indicates this file has 100 examples per class.

You could define additional flags if you wish. Possible candidates could be the batch size or the number of epochs.

```
def load_bottleneck_data(training_file, validation_file):
    """
    Utility function to load bottleneck features.

    Arguments:
        training_file - String
        validation_file - String
    """
    print("Training file", training_file)
    print("Validation file", validation_file)

    with open(training_file, 'rb') as f:
        train_data = pickle.load(f)
    with open(validation_file, 'rb') as f:
        validation_data = pickle.load(f)

    X_train = train_data['features']
    y_train = train_data['labels']
    X_val = validation_data['features']
    y_val = validation_data['labels']

    return X_train, y_train, X_val, y_va
```

A utility function that loads the bottleneck features from the pickled training and validation files.

Lab: Feature Extractio...

```
def main(_):
    # load bottleneck data
    X_train, y_train, X_val, y_val = load_bottleneck_data(FLAGS.train
ing_file, FLAGS.validation_file)

    print(X_train.shape, y_train.shape)
    print(X_val.shape, y_val.shape)

    # TODO: define your model and hyperparams here
    # make sure to adjust the number of classes based on
    # the dataset
    # 10 for cifar10
    # 43 for traffic

    # TODO: train your model here


# parses flags and calls the `main` function above
if __name__ == '__main__':
    tf.app.run()
```

This is where you'll define and train the model. Notice `FLAGS.training_file` and `FLAGS.validation_file` are passed into `load_bottleneck_data`. These refer to the *command line flags* defined earlier.

Once you've trained your model, record the results. How do they compare to the results from the previous exercise?

NEXT

Lab: Feature Extractio...