Advanced Techniques for CNNs and Keras

Data Augmentation

- One practical obstacle to building image classifiers is obtaining labeled training data.
- Finding images is difficult.
- Labeling images is time consuming and costly.
- How can me make the most out of the labelled data we have?

Data Augmentation

- If this is a chair:

Data Augmentation

- If this is a chair…

- Then so is this!

Data Augmentation

- If this is a chair…

- Also this:

Data Augmentation

- If this is a chair…

- Also this:

Data Augmentation

- By slightly altering images, we can increase our effective data size.
- Also allows the neural network to learn invariance to certain transformations.
- But we need to be careful – this can have unintended consequences.

Data Augmentation

Would not want a self-driving car to think these mean the same thing!

Data Flows in Keras

- Keras has a convenient mechanism for Data Augmentation.
- Requires use of “Data Generators”

- To date, we have used the standard `model.fit` mechanism
- Holds entire dataset in memory
- Reads the batches one by one out of memory

Data Flows in Keras

- Alternative mechanism is to use a “data generator”
- Idea: define a generator object which “serves” the batches of data.
- Then use `model.fit_generator` instead of `model.fit`
- Generators can be used to serve images from disk to conserve working memory

ImageDataGenerator

- Keras has an `ImageDataGenerator` class which permits “real-time” data-augmentation.
- When a batch of images is served, you can specify random changes to be made to the image.
- These include shifting, rotating, flipping, and various normalizations of the pixel values.

ImageDataGenerator

```
keras.preprocessing.image.ImageDataGenerator(
featurewise_center=False, samplewise_center=False,
featurewise_std_normalization=False, samplewise_std_normalization=False,
zca_whitening=False,
rotation_range=0.,
width_shift_range=0.,
height_shift_range=0.,
shear_range=0., zoom_range=0., channel_shift_range=0.,
fill_mode='nearest', cval=0.,
horizontal_flip=False, vertical_flip=False,
 rescale=None, preprocessing_function=None,
data_format=K.image_data_format())
```

Shifting Images

```
keras.preprocessing.image.ImageDataGenerator(
width_shift_range=0.,
height_shift_range=0.,
...)
```

```
width_shift_range=0.1
```

Shifting Images (how to fill in)

```
keras.preprocessing.image.ImageDataGenerator(
...,
fill_mode='nearest', cval=0.,
...)
```

```
cval "constant”, ”nearest”, “reflect”, “wrap”
```

Rotating Images

```
keras.preprocessing.image.ImageDataGenerator(
...,
rotation_range=0.,
...)
```

```
rotation_range=30
```

Flipping Images

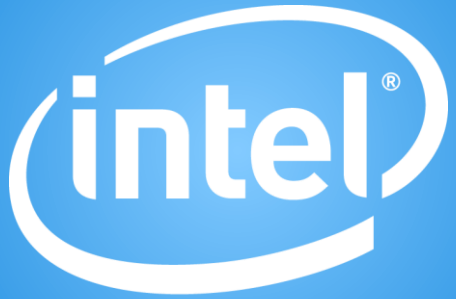
```
keras.preprocessing.image.ImageDataGenerator(
...,
horizontal_flip=False, vertical_flip=False,
...)
```

Keras Functional API

- So far we have primarily used the Keras Sequential method.
- Very convenient when each additional layer takes the results of the previous layer.
- However, suppose we have more than one input to a particular layer.
- Suppose we have multiple outputs, and want different loss functions for these outputs.
- More complicated graph structures require the Functional API.

Keras Functional API - principles

- Every layer is “callable” on a tensor, and returns a tensor.
- Specifically designate inputs using the `Input` layer.
- Merge outputs into a single output using `keras.layers.concatentate`
- Stack layers in a similar fashion to the Sequential model.
- Use `Model` to specify inputs and outputs of your model.
- When compiling, can specify different losses for different outputs, and specify how they should be weighted.



Software