Homework 5 Writeup

Instructions

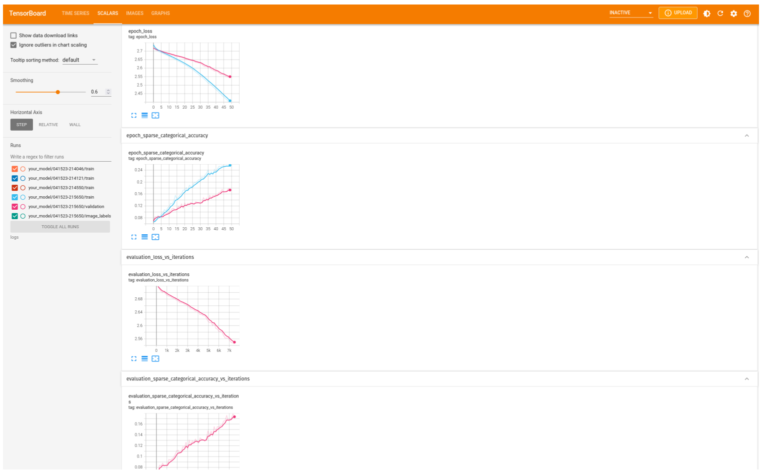
- Provide an overview about how your project functions.
- Describe any interesting decisions you made to write your algorithm.
- Show and discuss the results of your algorithm.
- Feel free to include code snippets, images, and equations.
- List any extra credit implementation and result (optional).
- Use as many pages as you need, but err on the short side.
- **Please make this document anonymous.**

Project Overview

This project is about training a convolutional neural network to perform scene classification. We train a head on a pre-trained model, and create and train a model from scratch.

Implementation Detail

Result



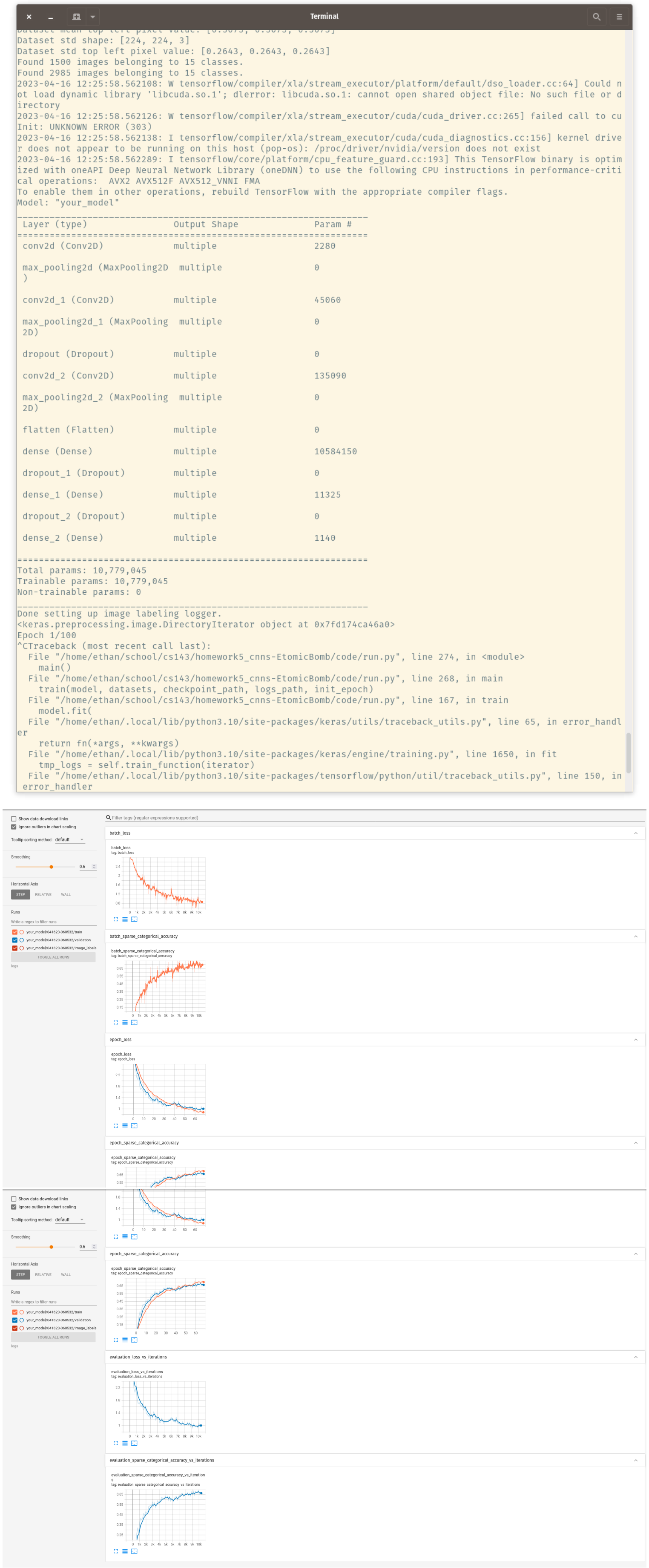
□ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP RELATIVE WALL

Runs
Write a regex to filter runs
☑ ○ your_model/041523-214046/train
☑ ○ your_model/041523-214121/train
☑ ○ your_model/041523-214550/train
☑ ○ your_model/041523-215650/train
☑ ○ your_model/041523-215650/validation
☑ ○ your_model/041523-215650/image_labels

Figure 1: Task 1:Training the small model on CPU. Somehow this was very fast.



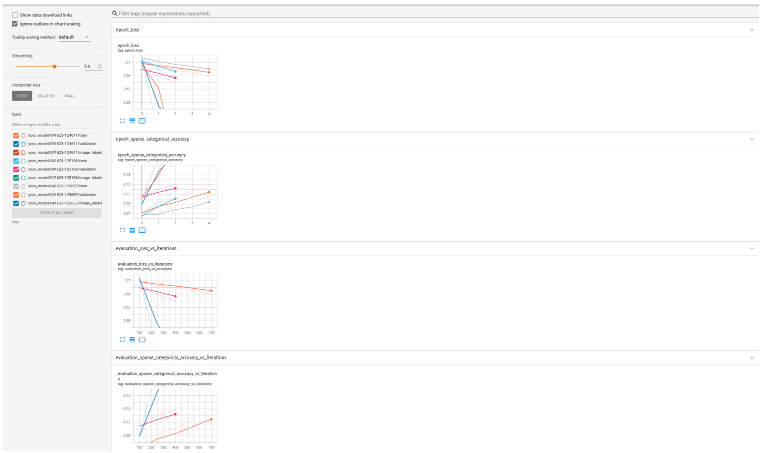
```
Dataset std shape: [224, 224, 3]
Dataset std top left pixel value: [0.2643, 0.2643, 0.2643]
Found 1500 images belonging to 15 classes.
Found 2985 images belonging to 15 classes.
2023-04-16 12:25:58.562108: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could n
ot load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open shared object file: No such file or d
irectory
2023-04-16 12:25:58.562126: W tensorflow/compiler/xla/stream_executor/cuda/cuda_driver.cc:265] failed call to cu
Init: UNKNOWN ERROR (303)
2023-04-16 12:25:58.562138: I tensorflow/compiler/xla/stream_executor/cuda/cuda_diagnostics.cc:156] kernel drive
r does not appear to be running on this host (pop-os): /proc/driver/nvidia/version does not exist
2023-04-16 12:25:58.562289: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optim
ized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-criti
cal operations:  AVX2 AVX512F AVX512_VNNI FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Model: "your_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             multiple                  2280

 max_pooling2d (MaxPooling2D  multiple                 0
 )

 conv2d_1 (Conv2D)           multiple                  45060

 max_pooling2d_1 (MaxPooling  multiple                 0
 2D)

 dropout (Dropout)           multiple                  0

 conv2d_2 (Conv2D)           multiple                  135090

 max_pooling2d_2 (MaxPooling  multiple                 0
 2D)

 flatten (Flatten)           multiple                  0

 dense (Dense)               multiple                  10584150

 dropout_1 (Dropout)         multiple                  0

 dense_1 (Dense)             multiple                  11325

 dropout_2 (Dropout)         multiple                  0

 dense_2 (Dense)             multiple                  1140

=================================================================
Total params: 10,779,045
Trainable params: 10,779,045
Non-trainable params: 0
_________________________________________________________________
Done setting up image labeling logger.
<keras.preprocessing.image.DirectoryIterator object at 0x7fd174ca46a0>
Epoch 1/100
^CTraceback (most recent call last):
  File "/home/ethan/school/cs143/homework5_cnns-EtomicBomb/code/run.py", line 274, in <module>
    main()
  File "/home/ethan/school/cs143/homework5_cnns-EtomicBomb/code/run.py", line 268, in main
    train(model, datasets, checkpoint_path, logs_path, init_epoch)
  File "/home/ethan/school/cs143/homework5_cnns-EtomicBomb/code/run.py", line 167, in train
    model.fit(
  File "/home/ethan/.local/lib/python3.10/site-packages/keras/utils/traceback_utils.py", line 65, in error_handl
er
    return fn(*args, **kwargs)
  File "/home/ethan/.local/lib/python3.10/site-packages/keras/engine/training.py", line 1650, in fit
    tmp_logs = self.train_function(iterator)
  File "/home/ethan/.local/lib/python3.10/site-packages/tensorflow/python/util/traceback_utils.py", line 150, in
 error_handler
```

□ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
☑ ○ your_model/041623-060532/train
☑ ○ your_model/041623-060532/validation
☑ ○ your_model/041623-060532/image_labels
TOGGLE ALL RUNS
logs
Filter tags (regular expressions supported)
batch_loss
tag: batch_loss
2.4
2
1.6
1.2
0.8
0 1k 2k 3k 4k 5k 6k 7k 8k 9k 10k
batch_sparse_categorical_accuracy
tag: batch_sparse_categorical_accuracy
0.65
0.55
0.45
0.35
0.25
0.15
0 1k 2k 3k 4k 5k 6k 7k 8k 9k 10k
epoch_loss
tag: epoch_loss
2.2
1.8
1.4
1
0 10 20 30 40 50 60
epoch_sparse_categorical_accuracy
tag: epoch_sparse_categorical_accuracy
0.65
0.55
0.45
0.35
0.25
0.15
0 10 20 30 40 50 60
evaluation_loss_vs_iterations
tag: evaluation_loss_vs_iterations
2.2
1.8
1.4
1
0 1k 2k 3k 4k 5k 6k 7k 8k 9k 10k
evaluation_sparse_categorical_accuracy_vs_iterations
tag: evaluation_sparse_categorical_accuracy_vs_iterations
0.65
0.55
0.45
0.35
0.25
0 1k 2k 3k 4k 5k 6k 7k 8k 9k 10k

Figure 2: Progress training the complete model

With	Epochs	Sparse categorization accuracy
Base	3	0.2315
Standardization	3	0.1300
Data augmentation	3	0.1008
Regularization	56	0.6873

Table 1: Task 1 performance after adding each intermediate sophistication. I was able to train the network with full standardization, augmentation, and regularization on Colab. The intermediate networks were trained on my laptop (which only has a cpu – thus limited epochs). This is because I was not able to access Colab GPUs due to availability limits when I went back to do at the intermediate stages. I believe the decreasing accuracy as more sophistication was added is just because of random chance and high variation from few epochs. I noticed that before adding data augmentation (on my lost intermediate run, when I still could use Colab) the training accuracy was able to get to basically 100, while the test accuracy stayed low - maybe in the 20s or 30s. When I added augmentation, the accuracy in training and testing coincided.



☐ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter runs

☑ your_model/041623-124611/train
☑ your_model/041623-124611/validation
☑ your_model/041623-124611/image_labels
☑ your_model/041623-125106/train
☑ your_model/041623-125106/validation
☑ your_model/041623-125106/image_labels
☑ your_model/041623-125657/train
☑ your_model/041623-125657/validation
☑ your_model/041623-125657/image_labels

TOGGLE ALL RUNS

logs

Filter tags (regular expressions supported)

epoch_loss
epoch_loss
tag: epoch_loss
2.7
2.66
2.62
2.58
0
1
2
3
4
epoch_sparse_categorical_accuracy
epoch_sparse_categorical_accuracy
tag: epoch_sparse_categorical_accuracy
0.15
0.13
0.11
0.09
0.07
0
1
2
3
4
evaluation_loss_vs_iterations
evaluation_loss_vs_iterations
tag: evaluation_loss_vs_iterations
2.7
2.66
2.62
2.58
150
250
350
450
550
650
750
evaluation_sparse_categorical_accuracy_vs_iterations
evaluation_sparse_categorical_accuracy_vs_iterations
tag: evaluation_sparse_categorical_accuracy_vs_iterations
0.15
0.13
0.11
0.09
150
250
350
450
550
650
750

Figure 3: Task 1: Progress training the intermediate models.

Type	Sparse categorization accuracy
Total	0.9106

Table 2: Task 3 performance

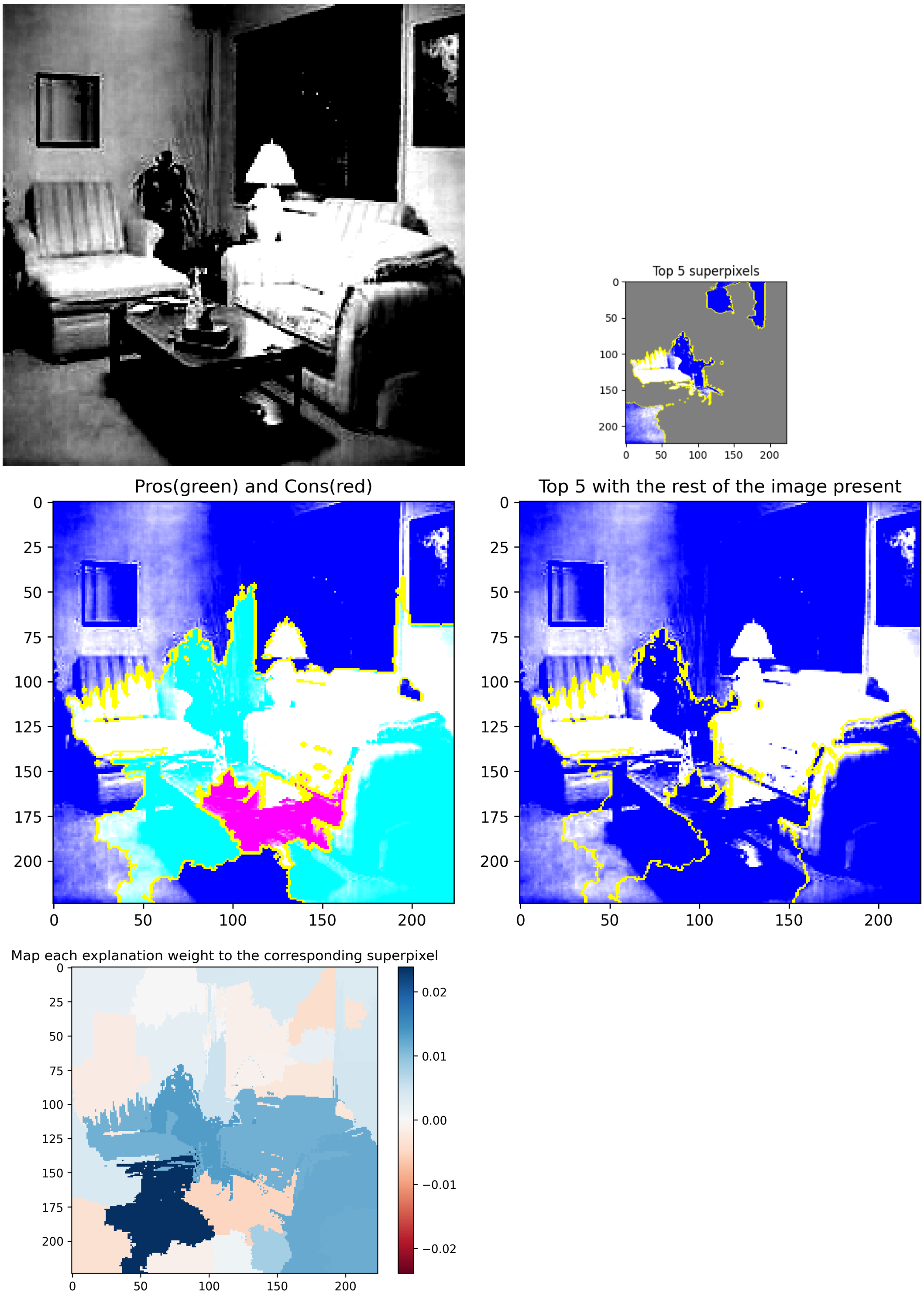


Figure 4: This is a bedroom, but was predicted as a living room. Curious. This model seemed to focus on the middle left of the image, especially the floor and part of the chair. It also seemed to attend to the window. These features – floors, chairs, and windows are all part of both bedrooms and living rooms. If it only attended to these features it's not suprising that it would get confused. I was not able to figure out whether it likes using the sky for its successful classifications.

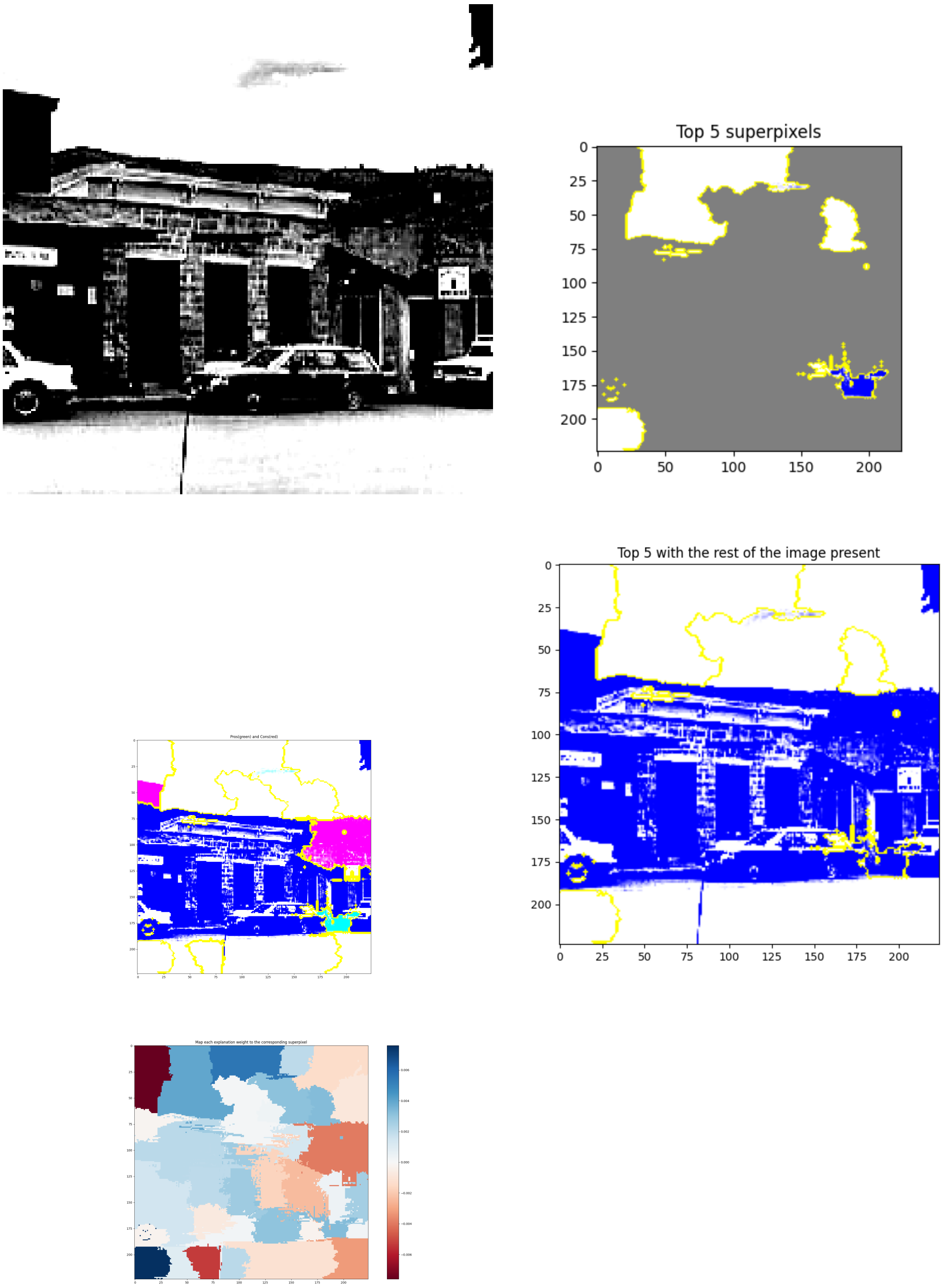
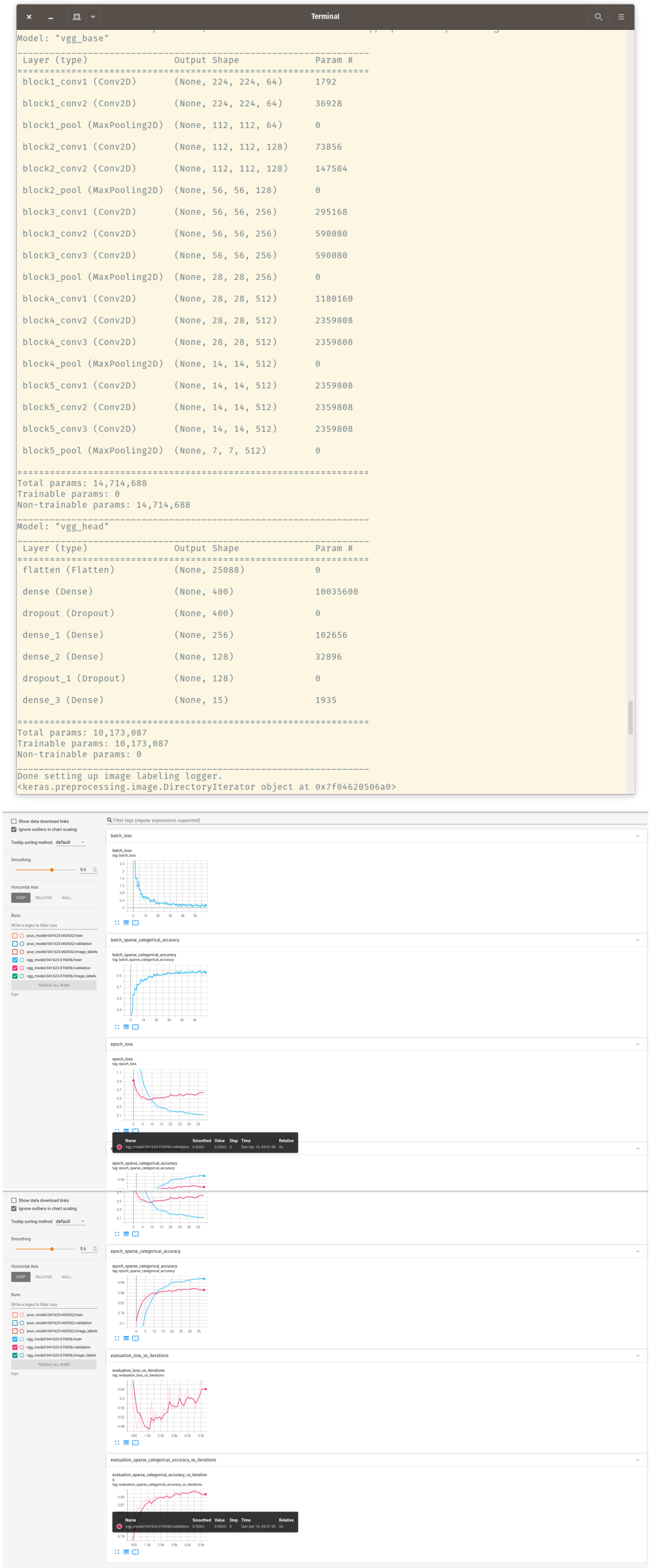


Figure 5: This is an inside city, but was predicted as a suburb. Curious. It seems like the model didn't really care about the architecture. Instead it focused on the sky and the front of a car, and part of the street. It would make sense that it would confuse an outdoor image of a suburb with a city if it was just looking at the sky.



Terminal

```
Model: "vgg_base"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
_________________________________________________________________
Model: "vgg_head"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 400)               10035600

 dropout (Dropout)           (None, 400)               0

 dense_1 (Dense)             (None, 256)               102656

 dense_2 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 15)                1935

=================================================================
Total params: 10,173,087
Trainable params: 10,173,087
Non-trainable params: 0
_________________________________________________________________
Done setting up image labeling logger.
<keras.preprocessing.image.DirectoryIterator object at 0x7f04620506a0>
```

☐ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing 0.6
Horizontal Axis: STEP
Runs
☐ ○ your_model/041623-060532/train
☐ ○ your_model/041623-060532/validation
☐ ○ your_model/041623-060532/image_labels
☑ ○ vgg_model/041623-070056/train
☑ ○ vgg_model/041623-070056/validation
☑ ○ vgg_model/041623-070056/image_labels

Name	Smoothed	Value	Step	Time	Relative
vgg_model/041623-070056/validation	0.9263	0.9263	0	Sun Apr 16, 03:01:50	0s

Filter tags (regular expressions supported)
RELATIVE
WALL
Write a regex to filter runs
TOGGLE ALL RUNS
logs
batch_loss
tag: batch_loss
2.4
2
1.6
1.2
0.8
0.4
0
1k
2k
3k
4k
5k
batch_sparse_categorical_accuracy
tag: batch_sparse_categorical_accuracy
0.9
0.7
0.5
0.3
epoch_loss
tag: epoch_loss
1.1
0.9
0.7
0.5
0.3
0.1
5
10
15
20
25
30
35
epoch_sparse_categorical_accuracy
tag: epoch_sparse_categorical_accuracy
0.94
0.88
0.82
0.76
0.7
evaluation_loss_vs_iterations
tag: evaluation_loss_vs_iterations
0.64
0.6
0.56
0.52
0.48
500
1.5k
2.5k
3.5k
4.5k
5.5k
evaluation_sparse_categorical_accuracy_vs_iterations
tag: evaluation_sparse_categorical_accuracy_vs_iterations
0.89
0.87
0.79

Figure 6: Task 3: Progress training head of the vgg model.