

CS 542 Class Challenge Report

Image Classification of COVID-19 X-rays

Ziqi Tan
U 88387934
E-mail: ziqi1756@bu.edu

April, 18, 2020

Contents

1	Introduction	2
2	Data Preprocessing and Data Augmentation	2
3	Task 1 [30 points]	2
3.1	Dataset	2
3.2	VGG16 Architecture for task 1	2
3.3	Training	3
3.4	Testing	3
3.4.1	Accuracy and Loss	4
3.4.2	t-SNE visualizations	4
4	Task 2 [30 points]	5
4.1	Dataset	5
4.2	VGG16 Architecture for task 2	5
4.3	Training	6
4.3.1	Parameters tuning	6
4.4	Testing	6
4.4.1	Accuracy and Loss	6
4.4.2	t-SNE visualizations	7
4.5	ResNet50V2 Architecture for task 2	7
4.6	Training	8
4.7	Testing	8
4.7.1	Accuracy and Loss	8
4.7.2	t-SNE visualization	9
4.8	Comparasion between VGG16 and ResNet50V2	9
5	Deploy Tasks on SCC Cluster [Bonus: 10 points]	10
5.1	Training time on CPU	10
5.2	Training time on GPU	11

1 Introduction

In this class challenge, we will classify X-ray images. The data we will use has been collected by Adrian Xu, combining the Kaggle Chest X-ray dataset with the COVID-19 Chest X-ray dataset collected by Dr. Joseph Paul Cohen of the University of Montreal. There are two folders: two that will be used for a binary classification task (Task1), and all that will be used for multi-class classification (Task2). An ipython notebook template is provided for each task.

The rest of this report will be organized as follows. Section 2 discusses the data preprocessing and data augmentation methods employed in these tasks. Section 3 and 4 provide solution for task 1 and task 2, respectively. Each task firstly introduces the model architecture and then list the optimizer, loss function, regularization and parameters. Finally, a t-Distributed Stochastic Neighbor Embedding (t-SNE) is used to reduce feature dimension. In task 1, a pre-train VGG16 performs well in a binary classification problem and it achieves a high accuracy of 95%, but a random guess test accuracy. Task 2 is a classification task with 4 classes. VGG16 as a good starter still achieves a high accuracy of 75%. Then, ResNet50V2 architecture is taken as a competing model, which can only achieves 45% validation accuracy and it gets a high test accuracy of 61.11%. Section 5 would be the deployment of the task running on the GPU of the SCC cluster.

2 Data Preprocessing and Data Augmentation

We could use `tf.keras.preprocessing.image.ImageDataGenerator` to preprocess the images and generate batches of tensor image data with real-time data augmentation.

- Rescaling the images ($rescale = 1.0/255$).
- Normalize the inputs to zero mean and divide by std of the dataset.
- Synthesize new images by rotating, shifting and flipping vertically and horizontally, zooming in and out and channel shift.
- Fraction of images reserved for validation: 0.2.

3 Task 1 [30 points]

Train a deep neural network model to classify normal vs. COVID-19 Xrays using the data in the folder two. Starting from a pre-trained model typically helps performance on a new task, e.g. starting with weights obtained by training on ImageNet. After training is complete, visualize features of training data by reducing their dimensionality to 2 using t-SNE. If your extracted features are good, data points representing a specific class should appear within a compact cluster.

3.1 Dataset

- Training set: 60 images of Covid and 70 of Normal.
- Test set: 9 images of Covid and 9 of Normal.

3.2 VGG16 Architecture for task 1

A pre-trained model VGG16 [1] has been employed in this architecture.

Architecture		
Layer (type)	Output shape	Number of parameters
Input	(None, 224, 224, 3)	0
VGG16 (Model)	(None, 7, 7, 512)	14714688
Flatten	(None, 25088)	0
Fully connected	(None, 256)	6422784
ReLU	(None, 256)	0
Fully connected	(None, 1)	257
Sigmoid	(None, 1)	0

- Total parameters: 21,137,729

- Trainable parameters: 6,423,041
- Non-trainable parameters: 14,714,688

VGG16 achieves a high accuracy on image classification. The 3 fully-connected layers at the top of the VGG16 network are re-defined as a flatten layer, a fully-connected layer with ReLu function and another fully-connected layer with Sigmoid function. ReLu performs well in Convolutional Neural Network. Thus, it is chosen as the activation for the first fully-connected layer.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 1: VGG16-Architecture [1]

3.3 Training

- Optimizer: Adam.
- Loss function: tf.keras.losses.BinaryCrossentropy for binary classification problem.
- Epochs: 40.
- Batch size: 10.

3.4 Testing

The validation accuracy achieves around 95%. However, the test accuracy is only 50% over a test set with size of 18, which is a random guess in a binary classification problem. The reason may be that the training data is not enough in the first place, secondly, the test data may have high bias.

3.4.1 Accuracy and Loss

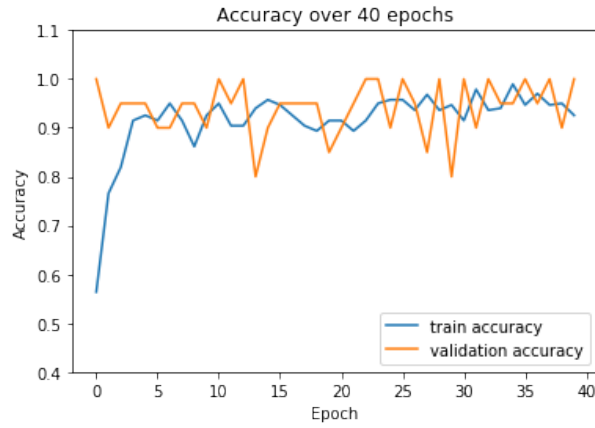


Figure 2: Accuracy on training set and validation set over 40 epochs

After 5 epochs, both train and validation accuracy becomes stable. Validation accuracy fluctuates around 95% more drastically than train accuracy does. The validation accuracy is approximately the same as the train accuracy, meaning that the model is not overfitting.

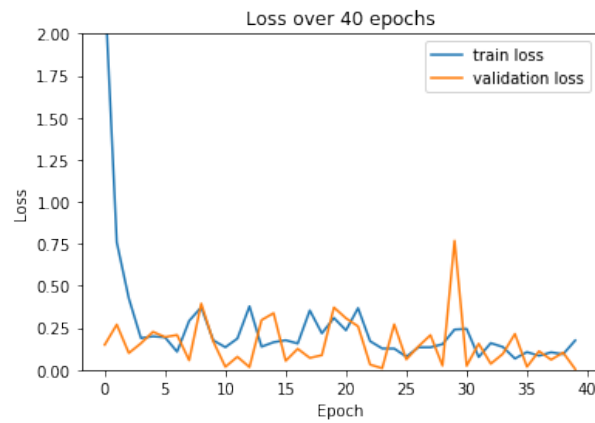


Figure 3: Loss on training set and validation set over 40 epochs

The train loss converges after 5 epochs, while the validation loss fluctuates and converges after 30 epochs.

3.4.2 t-SNE visualizations

We use the 130 training data to generate this feature distribution. The output of the first fully-connected layer with 256 neurons is taken as the high-dimension features. T-SEN is used to reduce 256 dimension features to 2 dimension.

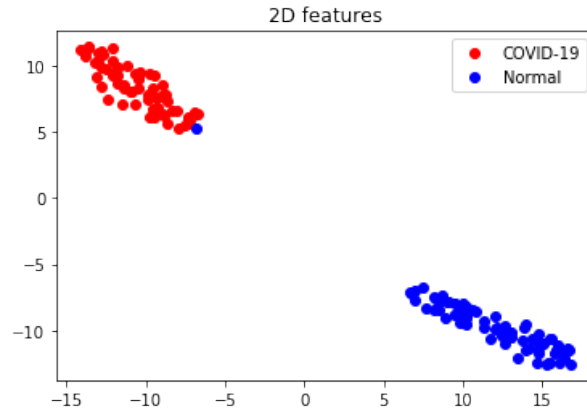


Figure 4: Two dimensional features on class COVID and Normal

The red points are COVID lung X-ray image and the blue are normal ones. The two classes is divide into two distinct clusters, meaning that a good classifier is established by the training of this VGG16 model.

4 Task 2 [30 points]

In this section you present your findings and results. Train a deep neural network model to classify an X-ray image into one of the following classes: normal, COVID-19, Pneumonia-Bacterial, and Pneumonia-Viral, using the folder all. Explore at least two different model architectures for this task, eg. AlexNet vs. VGG16. After training is complete, visualize features of training data by reducing their dimensionality to 2 using t-SNE. If your extracted features are good, data points representing a specific class should appear within a compact cluster.

4.1 Dataset

- Training set:

	COVID-19	Normal	Pneumonia-Bacterial	Pneumonia-Viral
Number of images	9	9	9	9

- Test set:

	COVID-19	Normal	Pneumonia-Bacterial	Pneumonia-Viral
Number of images	60	70	70	70

4.2 VGG16 Architecture for task 2

This is a subsection.

A pre-trained model VGG16 [1] has been employed in this architecture.

Architecture		
Layer (type)	Output shape	Number of parameters
Input	(None, 224, 224, 3)	0
VGG16 (Model)	(None, 7, 7, 512)	14714688
Flatten	(None, 25088)	0
Dropout1	(None, 25088)	0
Fully connected	(None, 2048)	51382272
ReLU	(None, 2048)	0
Dropout2	(None, 2048)	0
Fully connected	(None, 256)	524544
Fully connected	(None, 4)	1028
Softmax	(None, 4)	0

- Total parameters: 66,622,532
- Trainable parameters: 51,907,844
- Non-trainable parameters: 14,714,688

4.3 Training

- Optimizer: Adam.
- Loss function: `tf.keras.losses.CategoricalCrossentropy`.
- Epochs: 100.
- Regularization: Dropout is a regularization technique for reducing overfitting in neural networks. According to [1], dropout ratio is set to be 0.5 for each fully connected layer. In this case, we set 0.4 for dropout1 layer and 0.3 for dropout2.
- Batch size: 10.

4.3.1 Parameters tuning

Change batch size	
Batch size	Test accuracy
5	61%
6	69%
7	55%
8	47%
9	50%
10	75%
11	64%
12	77%
13	61%
14	61%
15	69%

4.4 Testing

Test accuracy is 75% and test loss is 0.7699.

4.4.1 Accuracy and Loss

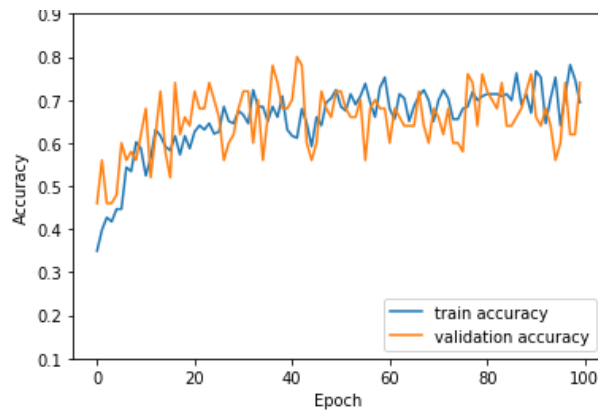


Figure 5: Accuracy on training set and validation set over 100 epochs

Both train accuracy and validation accuracy converge after 50 epochs. They fluctuate around 68%. Validation accuracy is slightly lower than train accuracy after 50 epochs, which means the model is not overfitting.

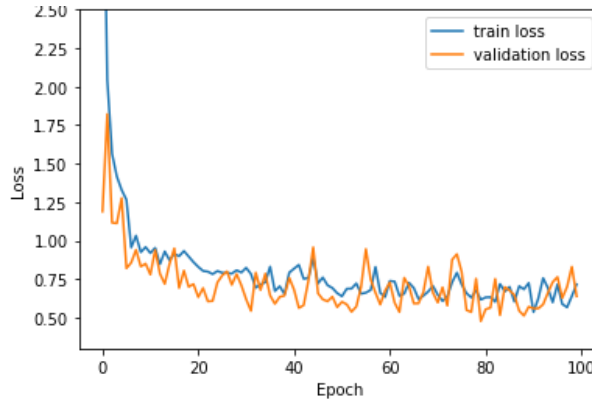


Figure 6: Loss on training set and validation set over 100 epochs

Both train loss and validation loss converge after 50 epochs.

4.4.2 t-SNE visualizations

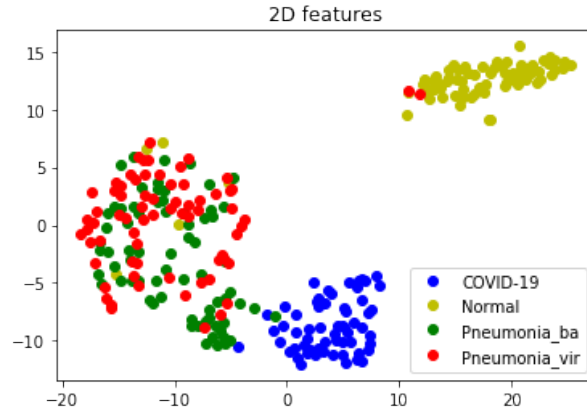


Figure 7: Two dimensional features on class COVID, Normal, Pneumonia_bacterial and Pneumonia_Viral

The four classes are clearly clustered in 3 groups. COVID-19 is the blue cluster and Normal is the yellow. Both Pneumonia_bacterial and Pneumonia_viral are in the red and green cluster, which means they have similar features.

4.5 ResNet50V2 Architecture for task 2

Architecture		
Layer (type)	Output shape	Number of parameters
Input	(None, 224, 224, 3)	0
ResNet50V2 (Model)	(None, 7, 7, 2048)	14714688
Average_pooling2d	(None, 1, 1, 2048)	0
Flatten	(None, 2048)	0
Dropout1	(None, 2048)	0
Fully connected	(None, 256)	524544
Dropout2	(None, 256)	0
Softmax	(None, 4)	1028

- Total parameters: 24,090,372
- Trainable parameters: 525,572
- Non-trainable parameters: 23,564,800

4.6 Training

- Optimizer: Adam with learning rate scheduler.
- Loss function: `tf.keras.losses.CategoricalCrossentropy`.
- Epochs: 100.
- Regularization: set 0.3 for dropout1 layer and 0.2 for dropout2.
- Batch size: 10.

Learning rate scheduler	
Epochs	Learning rate
(0,80)	1e-3
(80,120)	1e-1
(120,160)	1e-2
(160,180)	1e-3
Others	0.5e-3

4.7 Testing

The test accuracy: 61.11%.

4.7.1 Accuracy and Loss

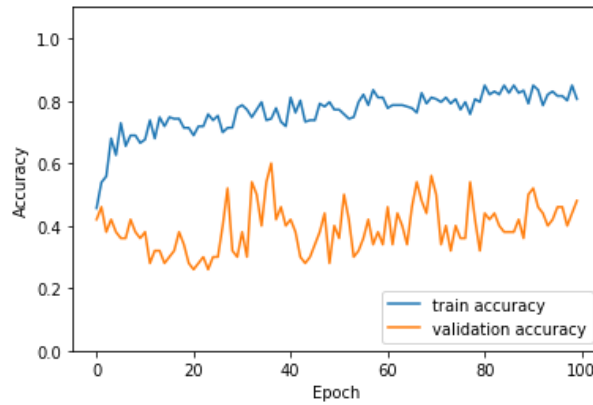


Figure 8: Accuracy on training set and validation set over 100 epochs

The train accuracy converges after 80 epochs on 78%, while the validation accuracy converges after 60 epochs on 40%. The train accuracy is about 35% higher than validation accuracy, indicating that an overfitting occurs, even though dropout regularization has been adapted. The reason may be that pre-trained model itself becomes overfitting.

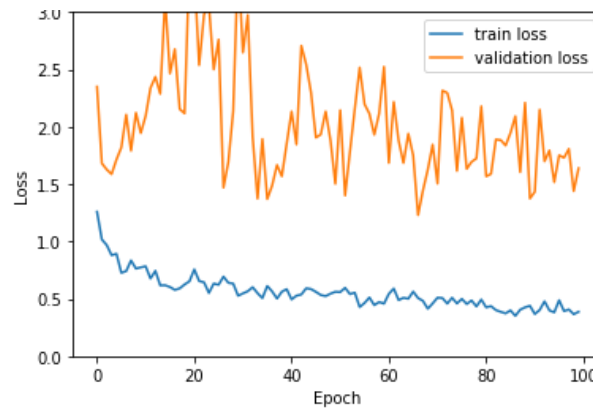


Figure 9: Loss on training set and validation set over 100 epochs

The validation loss is fairly higher than the train loss.

4.7.2 t-SNE visualization

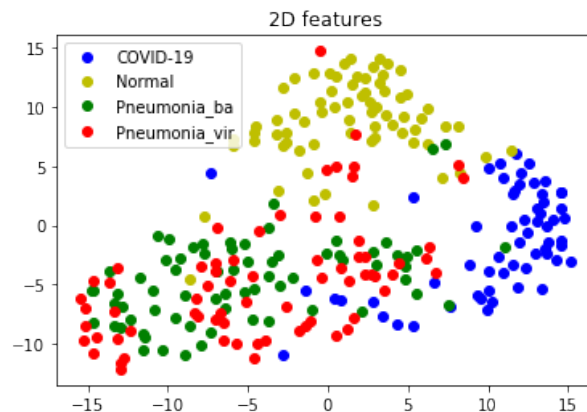


Figure 10: Two dimensional features on class COVID, Normal, Pneumonia_bacterial and Pneumonia_Viral

The four classes are clearly clustered in 3 groups. COVID-19 is the blue cluster and Normal is the yellow. Both Pneumonia_bacterial and Pneumonia_viral are in the red and green cluster, which means they have similar features.

4.8 Comparasion between VGG16 and ResNet50V2

Accuracy VGG16 outperforms ResNet50V2 mainly in two aspects: accuracy and model complexity. VGG16 achieve 75% test accuracy, while ResNet50V2 has only 61.11%.

Model complexity ResNet50V2 is more complex than VGG16.

Model Complexity		
Number of parameters	VGG16	ResNet50V2
Trainable	51,907,844	525,572
Non-Trainable	14,714,688	23,564,800
Total	66,622,532	24,090,372

- The number of non-trainable parameters of ResNet50V2 is about 1.5 times of that of VGG 16.
- A more complex model may be more likely to overfit the training data. The result shows that ResNet50V2 becomes overfitting while VGG16 does not, which may be caused by the over-complexity.

Training It is required to schedule the learning rate in ResNet50V2 to achieve higher accuracy. VGG16 only needs to use default Adam optimizer.

T-SEN VGG16 have a wider boundary between different clusters.

5 Deploy Tasks on SCC Cluster [Bonus: 10 points]

5.1 Training time on CPU

```
[name: "/device:CPU:0"  
device_type: "CPU"  
memory_limit: 268435456  
locality {  
}  
incarnation: 13504861637644514310  
, name: "/device:XLA_CPU:0"  
device_type: "XLA_CPU"  
memory_limit: 17179869184  
locality {  
}  
incarnation: 16759948233123524471  
physical_device_desc: "device: XLA_CPU device"  
]
```

Figure 11: The CPU library on SCC

Print the device library. We are assigned a CPU called "device:XLA_CPU:0".

```
21/21 [=====] - 27s 1s/step - loss: 0.2392 - accuracy: 0.8871 - val_loss: 0.3337 - val_accuracy: 0.8500  
Epoch 94/100  
21/21 [=====] - 28s 1s/step - loss: 0.2578 - accuracy: 0.8726 - val_loss: 0.2933 - val_accuracy: 0.8600  
Epoch 95/100  
21/21 [=====] - 28s 1s/step - loss: 0.2388 - accuracy: 0.8774 - val_loss: 0.3895 - val_accuracy: 0.7900  
Epoch 96/100  
21/21 [=====] - 27s 1s/step - loss: 0.2302 - accuracy: 0.8956 - val_loss: 0.3757 - val_accuracy: 0.8200  
Epoch 97/100  
21/21 [=====] - 29s 1s/step - loss: 0.2412 - accuracy: 0.8920 - val_loss: 0.3461 - val_accuracy: 0.8300  
Epoch 98/100  
21/21 [=====] - 28s 1s/step - loss: 0.2554 - accuracy: 0.8811 - val_loss: 0.2536 - val_accuracy: 0.8850  
Epoch 99/100  
21/21 [=====] - 28s 1s/step - loss: 0.2488 - accuracy: 0.8786 - val_loss: 0.3096 - val_accuracy: 0.8200  
Epoch 100/100  
21/21 [=====] - 27s 1s/step - loss: 0.2034 - accuracy: 0.8968 - val_loss: 0.3009 - val_accuracy: 0.8750  
/share/pkg.7/tensorflow/2.1.0/install/lib/SCC/./python3.6-avx/site-packages/keras_preprocessing/image/image_data_generator.py:71  
rator specifies 'featurewise_center', but it hasn't been fit on any training data. Fit it first by calling '.fit(numpy_data)'.  
warnings.warn('This ImageDataGenerator specifies '  
/share/pkg.7/tensorflow/2.1.0/install/lib/SCC/./python3.6-avx/site-packages/keras_preprocessing/image/image_data_generator.py:73  
rator specifies 'zca whitening', but it hasn't been fit on any training data. Fit it first by calling '.fit(numpy_data)'.  
warnings.warn('This ImageDataGenerator specifies '  
Training on CPU takes 2786  
[ziqil756@scc2 ziqil]$
```

Figure 12: The time we take when training on CPU

Training on CPU takes 2786 seconds as the Fig.12 shows.

5.2 Training time on GPU

```
zqi1756@scc1/projectnb/cs54: x +
scc-ondemand1.bu.edu pun/sys/shell/ssh/scc1/projectnb/cs542sb/zqi/
22
6
[{"name": "/device:CPU:0",
  device_type: "CPU",
  memory_limit: 268435456
}]
incarnation: 3220036562048926651
, name: "/device:XLA_GPU:0"
device_type: "XLA_GPU"
memory_limit: 17179869184
locality {
}
incarnation: 1162958223603516077
physical_device_desc: "device: XLA_CPU device"
, name: "/device:XLA_GPU:0"
device_type: "XLA_GPU"
memory_limit: 17179869184
locality {
}
incarnation: 11806237581770933406
physical_device_desc: "device: XLA_GPU device"
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 15675496858
locality {
  bus_id: 1
  links {
  }
}
}
incarnation: 9834201142839160068
physical_device_desc: "device: 0, name: Tesla V100-SXM2-16GB, pci bus id: 0000:1a:00.0, compute capability: 7.0"
2020-04-19 21:49:14.452807
Train for 21 steps, validate for 5 steps
Epoch 1/100
21/21 [=====] - 9s 413ms/step - loss: 2.1833 - accuracy: 0.6760 - val_loss: 1.2016 - val_accuracy: 0.7050
Epoch 2/100
21/21 [=====] - 5s 243ms/step - loss: 0.6752 - accuracy: 0.7464 - val_loss: 0.5864 - val_accuracy: 0.7500
Epoch 3/100
21/21 [=====] - 5s 235ms/step - loss: 0.7998 - accuracy: 0.7306 - val_loss: 0.6681 - val_accuracy: 0.7600
Epoch 4/100
21/21 [=====] - 5s 229ms/step - loss: 0.7278 - accuracy: 0.7670 - val_loss: 0.4971 - val_accuracy: 0.8550
Epoch 5/100
```

Figure 13: The GPU library on SCC

```
Epoch 96/100
21/21 [=====] - 5s 229ms/step - loss: 0.2077 - accuracy: 0.9005 - val_loss: 0.3156 - val_accuracy: 0.8300
Epoch 97/100
21/21 [=====] - 5s 221ms/step - loss: 0.2468 - accuracy: 0.8932 - val_loss: 0.2482 - val_accuracy: 0.8550
Epoch 98/100
21/21 [=====] - 5s 230ms/step - loss: 0.2476 - accuracy: 0.8857 - val_loss: 0.2697 - val_accuracy: 0.8750
Epoch 99/100
21/21 [=====] - 5s 230ms/step - loss: 0.2362 - accuracy: 0.8883 - val_loss: 0.2646 - val_accuracy: 0.8450
Epoch 100/100
21/21 [=====] - 5s 229ms/step - loss: 0.2299 - accuracy: 0.9000 - val_loss: 0.3358 - val_accuracy: 0.8150
/share/pkg.7/tensorflow/2.1.0/install/lib/SCC/./python3.6-gpu/site-packages/keras_preprocessing/image/image_data_generator.py:716: US
t it hasn't been fit on any training data. Fit it first by calling '.fit(numpy_data)'.
warnings.warn('This ImageDataGenerator specifies '
/share/pkg.7/tensorflow/2.1.0/install/lib/SCC/./python3.6-gpu/site-packages/keras_preprocessing/image/image_data_generator.py:735: US
hasn't been fit on any training data. Fit it first by calling '.fit(numpy_data)'.
warnings.warn('This ImageDataGenerator specifies '
Training on GPU takes 483
[zqi1756@scc1 zqi]$
```

Figure 14: The time we take when training on GPU

Training on GPU only takes 483 seconds.

References

- [1] Karen Simonyan and Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv 1409.1556, cs.cv, 2014
- [2] ResNet50V2 Documentation, https://keras.io/examples/cifar10_resnet/