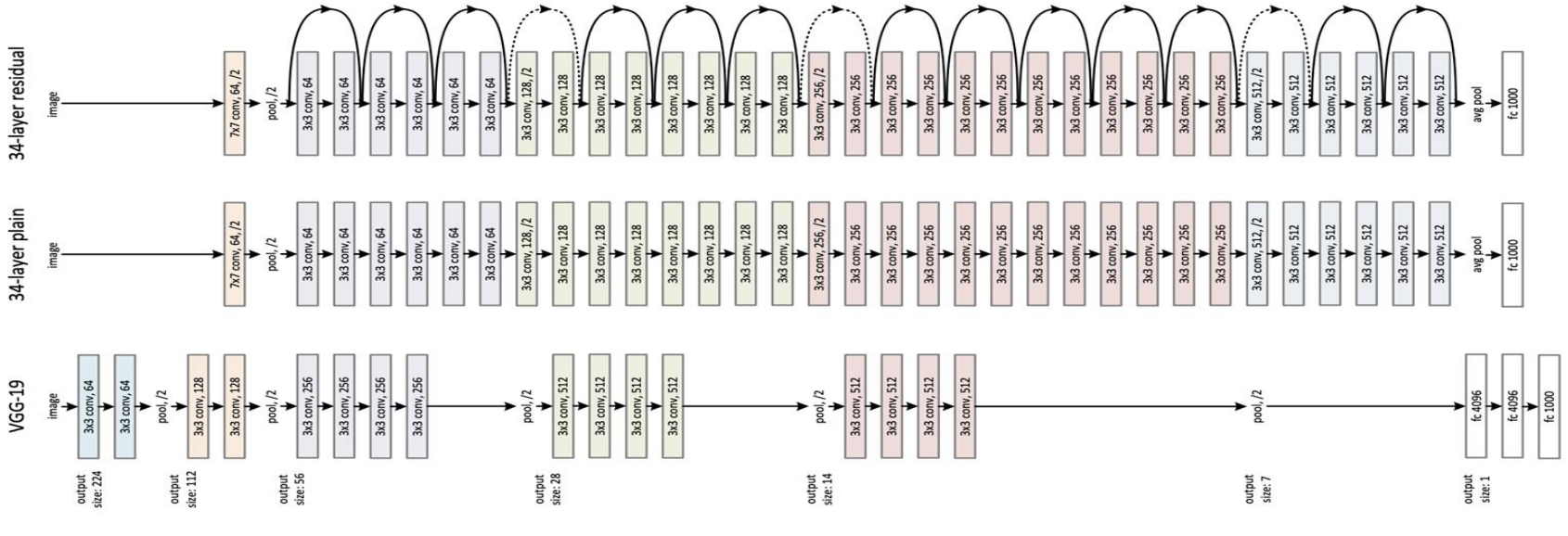
Residual Network Model (RNN) Exploration and Implementation

Presented by Yiru Xiong

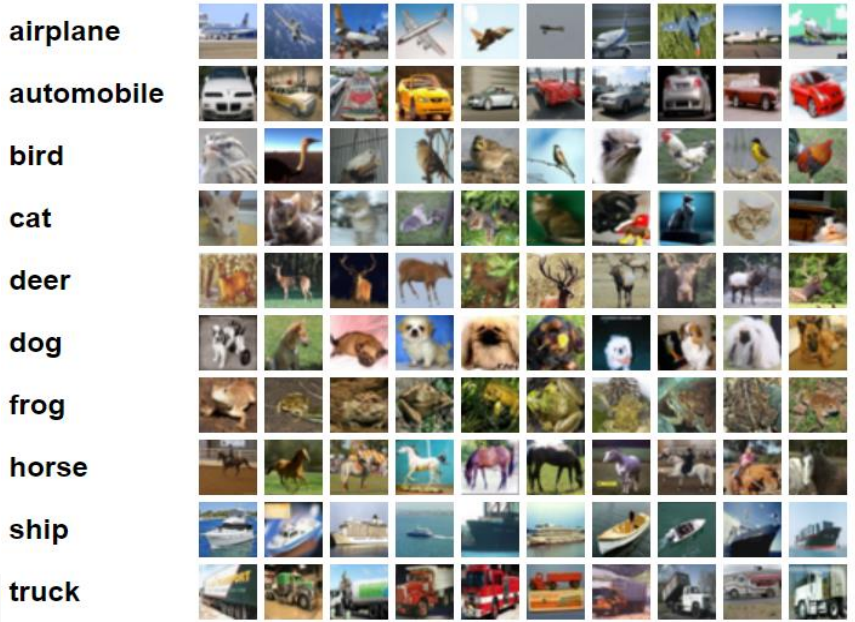
ResNet Recap

- A Convolutional Neural Network (CNN) based deep learning model
- Overcome Vanishing Gradient problem when increase the number of layers
- Skip connections connect activation of a layer to further layer while skipping some in between
- Stack residual blocks together
- Generally used for computer vision applications - image classifications, etc.



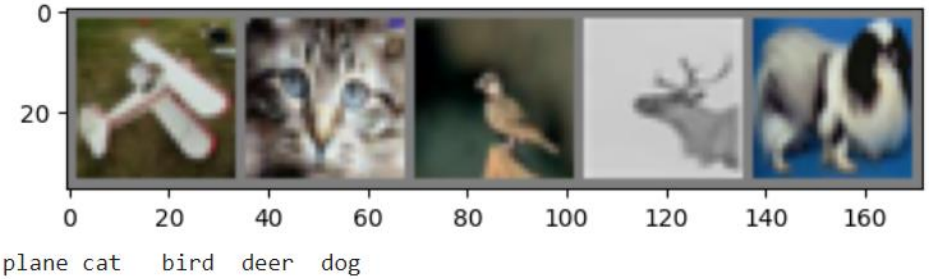
Data

- The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class.
- There are 50000 training images and 10000 test images.
- Classes are completely mutually exclusive.




```
# CIFAR-10 dataset
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

batch_size = 5

trainset = torchvision.datasets.CIFAR10(root='/content/drive/MyDrive/data', train=True, download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,shuffle=True, num_workers=2)

testset = torchvision.datasets.CIFAR10(root='/content/drive/MyDrive/data', train=False,download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=batch_size,shuffle=False, num_workers=2)

classes = ('plane', 'car', 'bird', 'cat','deer', 'dog', 'frog', 'horse', 'ship', 'truck')
```

Baseline Model



- A simple CNN model

```
==========================================================================
Layer (type:depth-idx)                                   Param #
==========================================================================
CNN                                                      --
├─Conv2d: 1-1                                            456
├─MaxPool2d: 1-2                                         --
├─Conv2d: 1-3                                            2,416
├─Linear: 1-4                                            48,120
├─Linear: 1-5                                            10,164
├─Linear: 1-6                                            850
==========================================================================
Total params: 62,006
Trainable params: 62,006
Non-trainable params: 0
==========================================================================
```

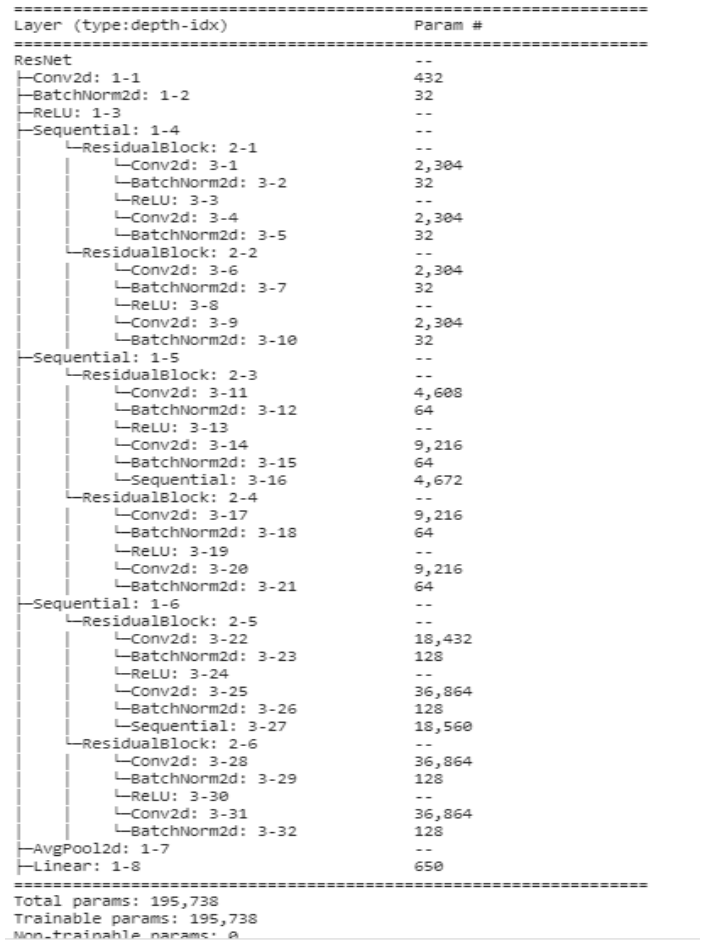
- Optimizer: SGD
- Learning rate: 0.001
- Momentum: 0.9
- Epoch_num: 10

Accuracy on test images: 62%

ResNet

- Optimizer: SGD
- Learning rate: 0.001
- Epoch_num: 10
- With learning rate decay

Accuracy on test images: 80.95%



```
==========================================================================
Layer (type:depth-idx)                   Param #
==========================================================================
ResNet                                   --
├─Conv2d: 1-1                            432
├─BatchNorm2d: 1-2                       32
├─ReLU: 1-3                              --
├─Sequential: 1-4                        --
│    └─ResidualBlock: 2-1                --
│    │    └─Conv2d: 3-1                  2,304
│    │    └─BatchNorm2d: 3-2             32
│    │    └─ReLU: 3-3                    --
│    │    └─Conv2d: 3-4                  2,304
│    │    └─BatchNorm2d: 3-5             32
│    └─ResidualBlock: 2-2                --
│    │    └─Conv2d: 3-6                  2,304
│    │    └─BatchNorm2d: 3-7             32
│    │    └─ReLU: 3-8                    --
│    │    └─Conv2d: 3-9                  2,304
│    │    └─BatchNorm2d: 3-10            32
├─Sequential: 1-5                        --
│    └─ResidualBlock: 2-3                --
│    │    └─Conv2d: 3-11                 4,608
│    │    └─BatchNorm2d: 3-12            64
│    │    └─ReLU: 3-13                   --
│    │    └─Conv2d: 3-14                 9,216
│    │    └─BatchNorm2d: 3-15            64
│    │    └─Sequential: 3-16             4,672
│    └─ResidualBlock: 2-4                --
│    │    └─Conv2d: 3-17                 9,216
│    │    └─BatchNorm2d: 3-18            64
│    │    └─ReLU: 3-19                   --
│    │    └─Conv2d: 3-20                 9,216
│    │    └─BatchNorm2d: 3-21            64
├─Sequential: 1-6                        --
│    └─ResidualBlock: 2-5                --
│    │    └─Conv2d: 3-22                 18,432
│    │    └─BatchNorm2d: 3-23            128
│    │    └─ReLU: 3-24                   --
│    │    └─Conv2d: 3-25                 36,864
│    │    └─BatchNorm2d: 3-26            128
│    │    └─Sequential: 3-27             18,560
│    └─ResidualBlock: 2-6                --
│    │    └─Conv2d: 3-28                 36,864
│    │    └─BatchNorm2d: 3-29            128
│    │    └─ReLU: 3-30                   --
│    │    └─Conv2d: 3-31                 36,864
│    │    └─BatchNorm2d: 3-32            128
├─AvgPool2d: 1-7                         --
├─Linear: 1-8                            650
==========================================================================
Total params: 195,738
Trainable params: 195,738
Non-trainable params: 0
```

ResNet - Hyperparameter Tuning

Grid Search -

- Batch Size - 128
- Number of epochs - 20
- Learning rate - 0.001
- Optimization algorithm - Adam
- With/ without learning rate decay -with

Accuracy on test images: 86.18%

```
# optimization algorithm
optim_grid = [optim.SGD, optim.RMSprop, optim.Adagrad, optim.Adadelta,optim.Adam, optim.Adamax, optim.NAdam]
lr_grid = [0.001,0.005, 0.007, 0.01, 0.05]
epochs_grid = [10, 30, 50, 80]
batch_grid = [100, 150, 200, 500]
```

ResNet - Continued

Adding dropout layer

- Dropout layer can be added to visible layer
- Dropout layer can be added to hidden layers
- Dropout rate

  0.2 - meaning one in five inputs will be

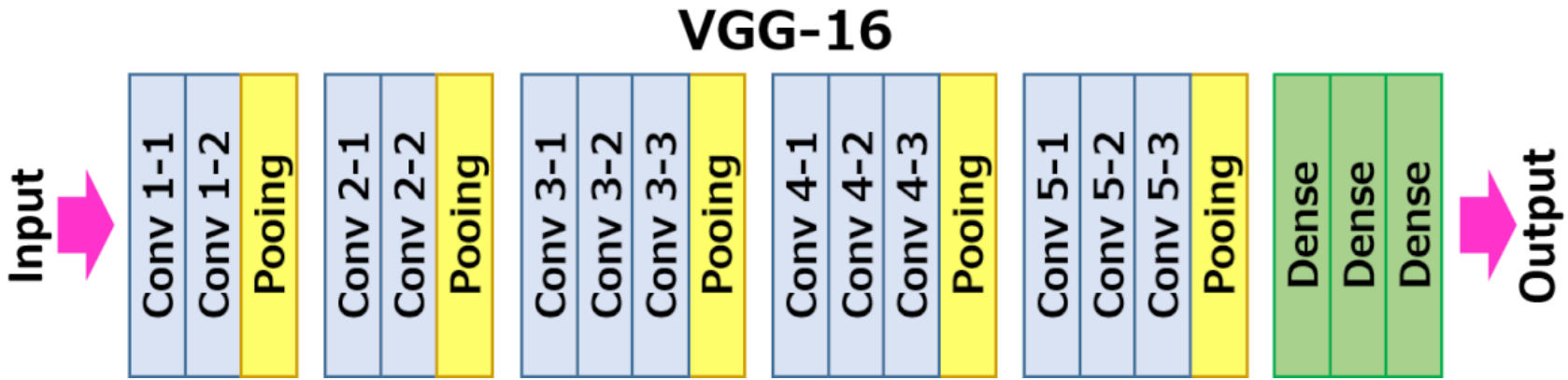
  randomly excluded from each update cycle
- Applied after the activation layer

Accuracy on test images: avg 86%

```
==========================================================================
Layer (type:depth-idx)                   Param #
==========================================================================
ResNet                                   --
├─Conv2d: 1-1                            432
├─BatchNorm2d: 1-2                       32
├─ReLU: 1-3                              --
├─Sequential: 1-4                        --
│    └─ResidualBlock: 2-1                --
│    │    └─Conv2d: 3-1                  2,304
│    │    └─BatchNorm2d: 3-2             32
│    │    └─ReLU: 3-3                    --
│    │    └─Conv2d: 3-4                  2,304
│    │    └─BatchNorm2d: 3-5             32
│    └─ResidualBlock: 2-2                --
│    │    └─Conv2d: 3-6                  2,304
│    │    └─BatchNorm2d: 3-7             32
│    │    └─ReLU: 3-8                    --
│    │    └─Conv2d: 3-9                  2,304
│    │    └─BatchNorm2d: 3-10            32
├─Sequential: 1-5                        --
│    └─ResidualBlock: 2-3                --
│    │    └─Conv2d: 3-11                 4,608
│    │    └─BatchNorm2d: 3-12            64
│    │    └─ReLU: 3-13                   --
│    │    └─Conv2d: 3-14                 9,216
│    │    └─BatchNorm2d: 3-15            64
│    │    └─Sequential: 3-16             4,672
│    └─ResidualBlock: 2-4                --
│    │    └─Conv2d: 3-17                 9,216
│    │    └─BatchNorm2d: 3-18            64
│    │    └─ReLU: 3-19                   --
│    │    └─Conv2d: 3-20                 9,216
│    │    └─BatchNorm2d: 3-21            64
├─Sequential: 1-6                        --
│    └─ResidualBlock: 2-5                --
│    │    └─Conv2d: 3-22                 18,432
│    │    └─BatchNorm2d: 3-23            128
│    │    └─ReLU: 3-24                   --
│    │    └─Conv2d: 3-25                 36,864
│    │    └─BatchNorm2d: 3-26            128
│    │    └─Sequential: 3-27             18,560
│    └─ResidualBlock: 2-6                --
│    │    └─Conv2d: 3-28                 36,864
│    │    └─BatchNorm2d: 3-29            128
│    │    └─ReLU: 3-30                   --
│    │    └─Conv2d: 3-31                 36,864
│    │    └─BatchNorm2d: 3-32            128
├─AvgPool2d: 1-7                         --
├─Dropout: 1-8                           --
├─Linear: 1-9                            650
==========================================================================
Total params: 195,738
Trainable params: 195,738
Non-trainable params: 0
==========================================================================
```

Transfer Learning

- Leveraging pre-trained high-performance model
- VGG-16: Very Deep Convolutional Network for Large-scale Image Classification trained on ImageNet
- Convolutional Layers = 13, Pooling Layers = 5, Dense Layers = 3



Transfer Learning - continued

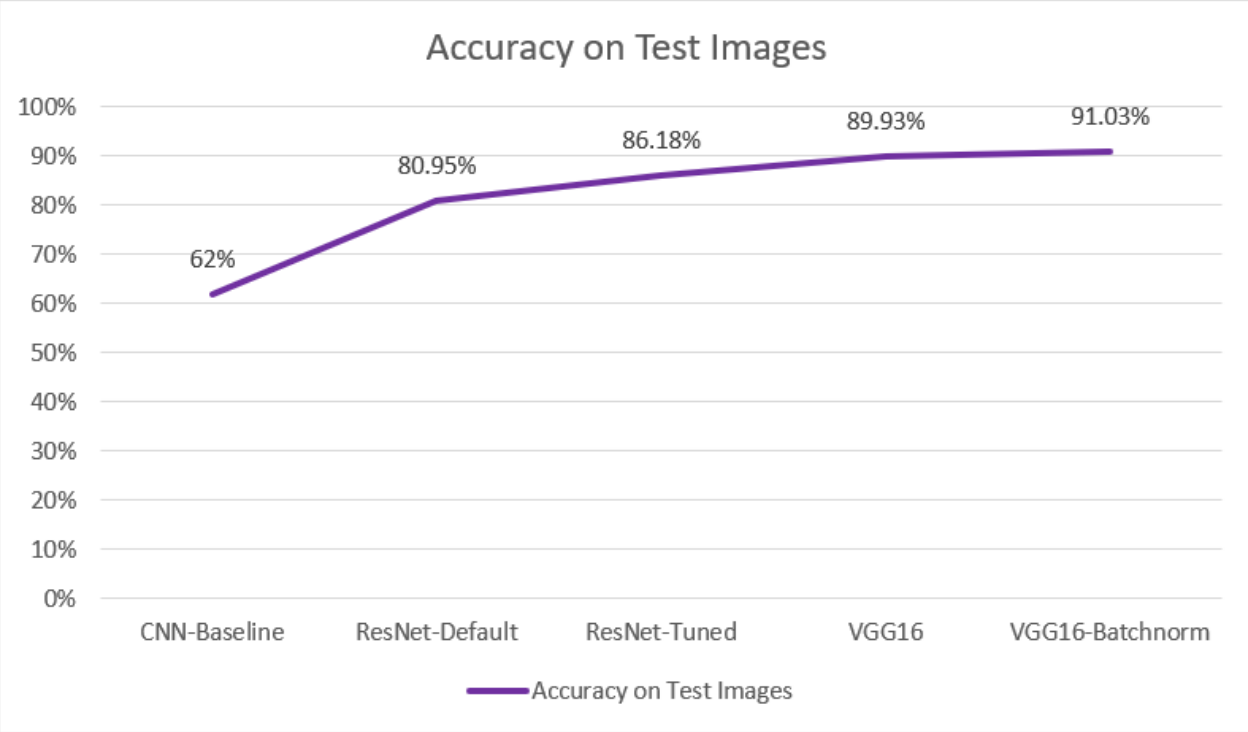
- VGG-16
- VGG-16 with batch normalization
- Experiment with different learning rate 0.001, 0.003

  Num_epochs: 10, 20

  Optimization Algorithm: Adam, RMSprop
- With the same parameter settings, adding batch normalization boosts the overall accuracy

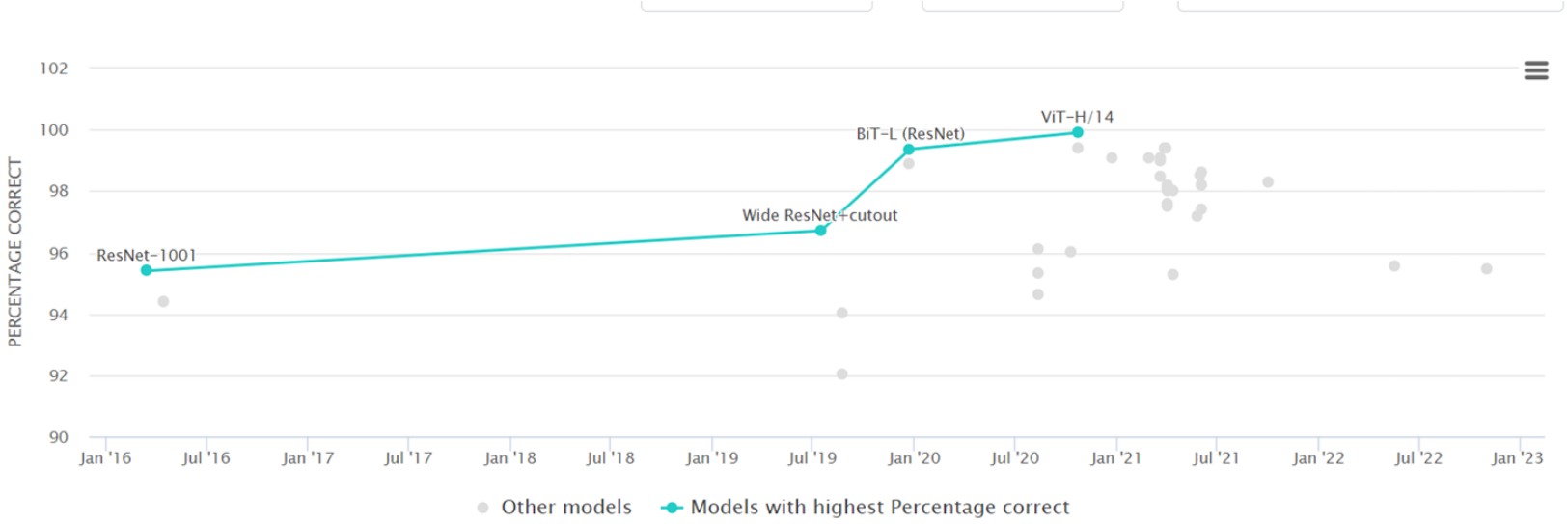
- **Best Results:**
  VGG-16: Accuracy on Testing Images: 89.93
  VGG-16 with batch normalization: Accuracy on testing images; 91.03%

Experiment Results



Future Work

1. Vit - Vision Transformer. Attention Neural Network Architecture



2. Automatic Hyperparameter Tuning - Optuna

References

1. Tam, A. (2023, April 7). How to grid search hyperparameters for PyTorch Models. MachineLearningMastery.com.https://machinelearningmastery.com/how-to-grid-search-hyperparameters-for-pytorch-models/

1. GitHub,github.com/rasbt/deeplearning-models/blob/master/pytorch_ipynb/transfer/transferlearning-vgg16-cifar10-1.ipynb. Accessed 21 Nov. 2023.

1. “Accelerate Your Hyperparameter Optimization with PyTorch’s Ecosystem Tools.” Medium, 14 Sept. 2020, medium.com/pytorch/accelerate-your-hyperparameter-optimization-with-pytorchs-ecosystem-tools-bc17001b9a49.