UDACITY

Transfer Learning

Lamia Zain
lamiahasan4@gmail.com
LinkedIn

July, 05, 2024

Connect Sessions | Purpose

A Connect Session **IS**:

- Focused on learning, encouragement & graduation for a group of students coached by a Udacity Session Lead
- Setting weekly study goals
- Helping each other with progress (including peer to peer)
- Keeping everyone accountable for their responsibilities
- A way to meet individuals in tech field & learn about the industry
- **Mandatory**

A Connect Session **IS NOT**:

- A social meetup
- A study group
- A substitute for online learning
- **Optional**

UDACITY

Let’s check your progress

You are encouraged to spend at lest 10 hours/week to graduate.

Presentation date

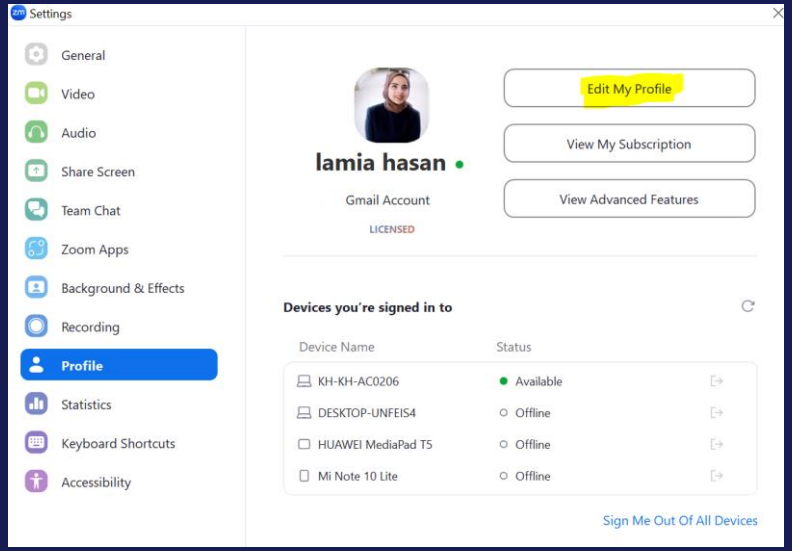
UDACITY

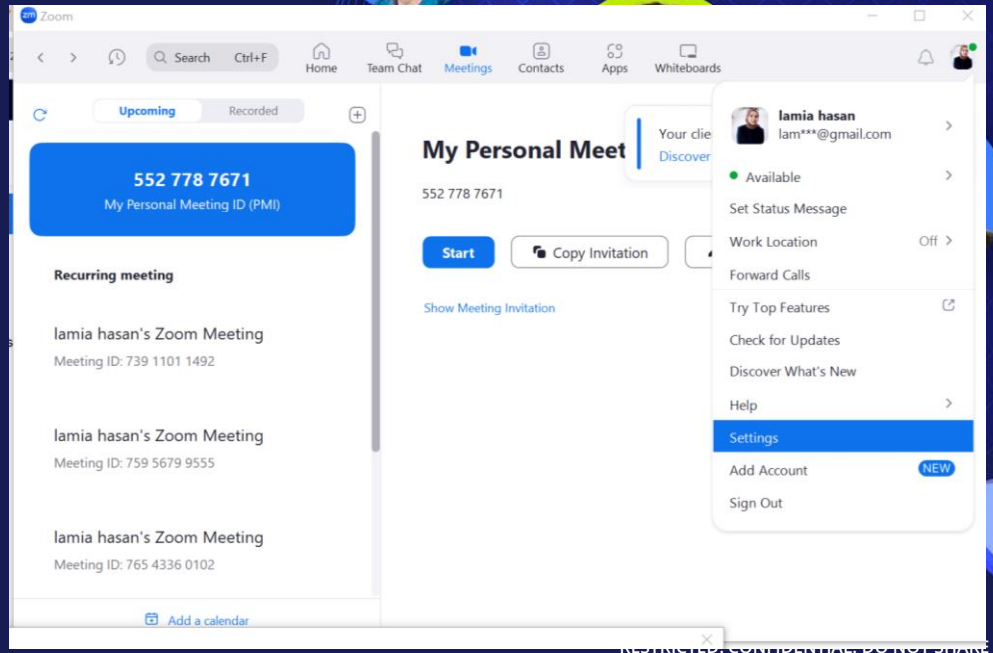
Attendance is taken automatically

Please change your name to be First Name and Last name on Zoom
Like : Lamia Zain

UDACITY

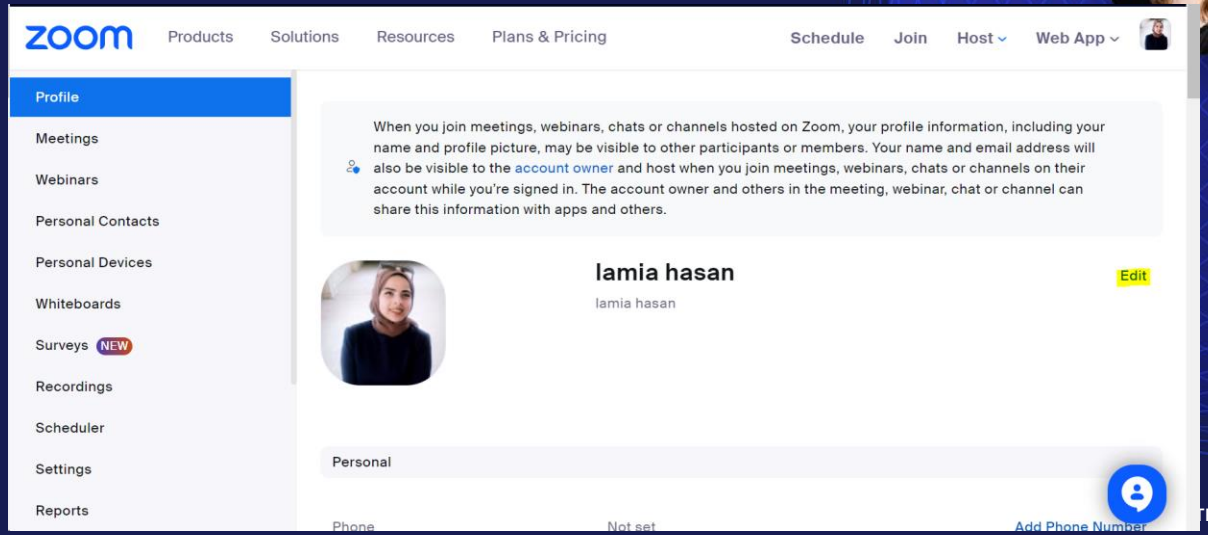
Change your Name on Zoom





UDACITY

Change your Name on Zoom



Session Lead role:

Communication Chart

Issue	Where to go?
Classroom access/ Withdrawal/ Graduation issues/ Plagiarism/ Project Review Inquiries	Email support@udacity.com
Technical Issues, Attendance, Content Related Issues/ Project inquiries	Session Lead
Session Switch/ Community related issues	Community Moderators

Program Timeline & Learning Journey

2024

April | May | June | July | August

Program Kickoff
April 10, 2024

Revoking Deadline
May 12, 2024

End of Program
August 28, 2024

First Project Submission
May 1, 2024

Second Project Submission
June 5, 2024

Third Project Submission
July 10, 2024

Fourth Project Submission
August 21, 2024

Program Period
April 10 - August 28, 2024

Four-weeks Agenda, Weekly schedule

Week 10	Jun 12, 2024			Finish the lessons below from **the Convolutional Neural Networks** Introduction to CNNs CNN Concepts **[Work on/submit the #3 project: Landmark Classification & Tagging for Social Media]**	Convolutional Neural Networks Introduction to CNNs CNN Concepts
Week 11	Jun 19, 2024			Finish the lessons below from **the Convolutional Neural Networks** CNNs in Depth **[Work on/submit the #3 project: Landmark Classification & Tagging for Social Media]**	Convolutional Neural Networks CNNs in Depth
Week 12	Jun 26, 2024			Finish the lessons below from **the Convolutional Neural Networks** Transfer Learning **[Work on/submit the #3 project: Landmark Classification & Tagging for Social Media]**	Convolutional Neural Networks Transfer Learning
Week 13	Jul 3, 2024			Finish the lessons below from **the Convolutional Neural Networks** Autoencoders **[Work on/submit the #3 project: Landmark Classification & Tagging for Social Media]**	Convolutional Neural Networks Autoencoders Project Walkthrough: Landmark Classification & Tagging for Social Media
Week 14	Jul 10, 2024	Jul 10, 2024	**Landmark Classification & Tagging for Social Media**	Finish the lessons below from **the Convolutional Neural Networks** Object Detection and Segmentation **[Work on/submit the #3 project: Landmark Classification & Tagging for Social Media]**	Convolutional Neural Networks Object Detection and Segmentation Project Walkthrough: Landmark Classification & Tagging for Social Media

Student Milestone | Revoking

REVOKING

Revoking is the process by which Udacity removes a student from a Nanodegree program.

AWS reserves the right to revoke you from the program if you do not comply with program requirements.

CRITERIA

Students can be revoked if they fail to:

- Submit Project 1
- Complete the required concepts

The date of **Revoking** for the scholarship is **May 12, 2024**

Code of Conduct | Plagiarism

BASIC RULES

- Project submissions must consist of original work
- Submitted projects will be scanned for plagiarism
- Students who are found to have plagiarised will risk their Nanodegree being revoked
- Read the honor code and the rubric carefully for all projects

Recap

- Batch Normalization
- Data Augmentation
- Train a Custom CNN model with the Tiny ImageNet Dataset From HuggingFace

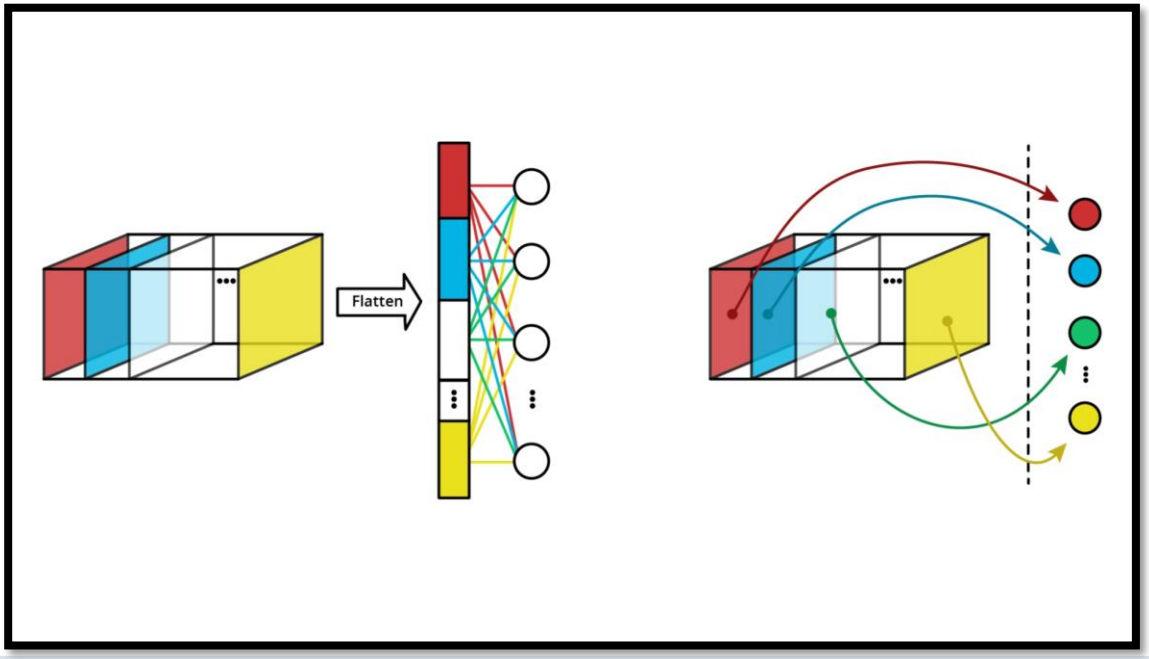
Objectives:

- GAP Layers
- Attention Layers
- Transfer Learning
- Use a Pretrained model with the Tiny ImageNet Dataset From HuggingFace

Global Average Booling Layers

GAP Layers

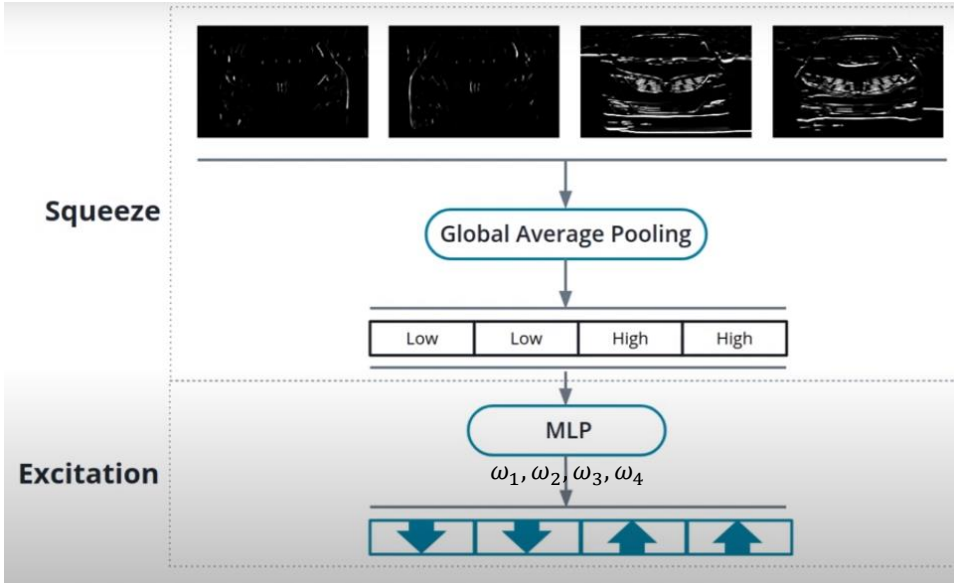
- Squeeze (GAP Layer) -> Generates Weights corresponding to the number of feature maps.
- Size of GAP Layers won't be affected by the size of the input Images, It depends only on the number of feature maps coming out of the final Convolutional Layer.



Attention Layer

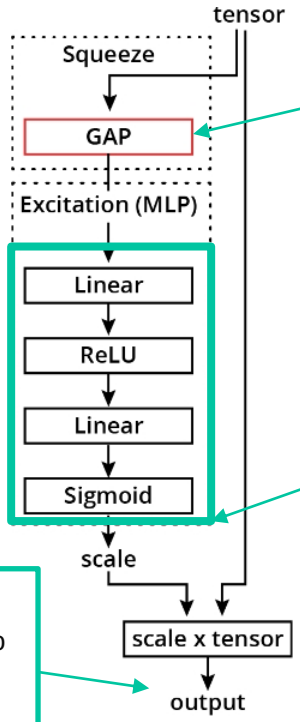
Attention Layers:

- Paying attention to sum feature maps more than the others.
- By using Squeeze and Excitation technique
- Squeeze (GAP Layer) -> Generates Weights corresponding to the number of feature maps.



Attention Layers:

- Attention Layers are applied after each convolution Layer



Convert M Feature maps to M weights

Vector of weights ranging from 0 to 1. Size Of the Output Vector = Size Of the Input Vector

Multiply by the input feature maps again to determine the significance / insignificance of any

SE Block in PyTorch:

Add SE Block

```
[ ] class SEBlock(nn.Module):
        def __init__(self,in_channels):
            super(SEBlock, self).__init__()
            self.global_avg_pool = nn.AdaptiveAvgPool2d(1) #Averaging the pixels in one channel to give one final output
            self.fc1 = nn.Linear(in_channels,in_channels)  #n.channels -> 8

        def forward(self,x):
            se_tensor = self.global_avg_pool(x)                              # (Batches, Channels)
            se_tensor = se_tensor.view(se_tensor.size(0), -1)                #Put in 1D array
            se_tensor = torch.sigmoid(F.relu(self.fc1(se_tensor))) #(Batches, Channels)
            # Reshape to (batch_size, channels, 1, 1) to multiply by input
            se_tensor = se_tensor.view(se_tensor.size(0), se_tensor.size(1), 1, 1)
            # Multiply the original feature maps by the SE tensor
            x = x * se_tensor
            return x
```

Define the SE Block

```
def forward(self, x):

    x = self.bn1(F.relu(self.conv1(x)))
    x = self.se_block1(x)
    x = self.pool(x)

    x = self.bn2(F.relu(self.conv2(x)))
    x = self.se_block2(x)
    x = self.drop(x)
```

Calling the SE block in the Forward Method

```
class MyCNN(nn.Module):
    def __init__(self):
        # We optimize dropout rate in a convolutional neural network.
        super(MyCNN, self).__init__()

        self.conv1 = nn.Conv2d(in_channels = 3, out_channels=16, kernel_size=3, stride=1, padding= 1)
        self.bn1 = nn.BatchNorm2d(16)
        self.se_block1 = SEBlock(in_channels=16)


        self.conv2 = nn.Conv2d(in_channels=16,out_channels=32,kernel_size=3,stride=1,padding=1)
        self.bn2 = nn.BatchNorm2d(32)
        self.se_block2 = SEBlock(in_channels=32)
```

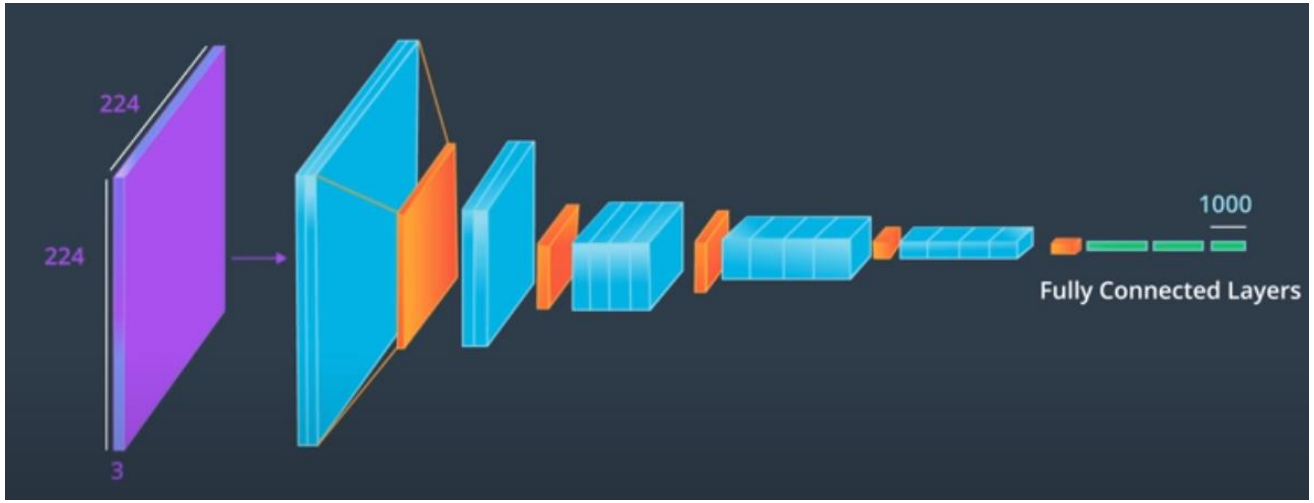
Initializing the SE block after the convolution Layer

Let's apply to Tiny ImageNet

Transfer learning

Why?

- Best transfer knowledge gained from training huge networks on huge dataset to our own problem.
- Ex: Classify 10 classes of CIFAR10 dataset using a network like RESNET18 / RESNET50 that was trained on ImageNet Dataset that contained millions of images with 1000 output classes.

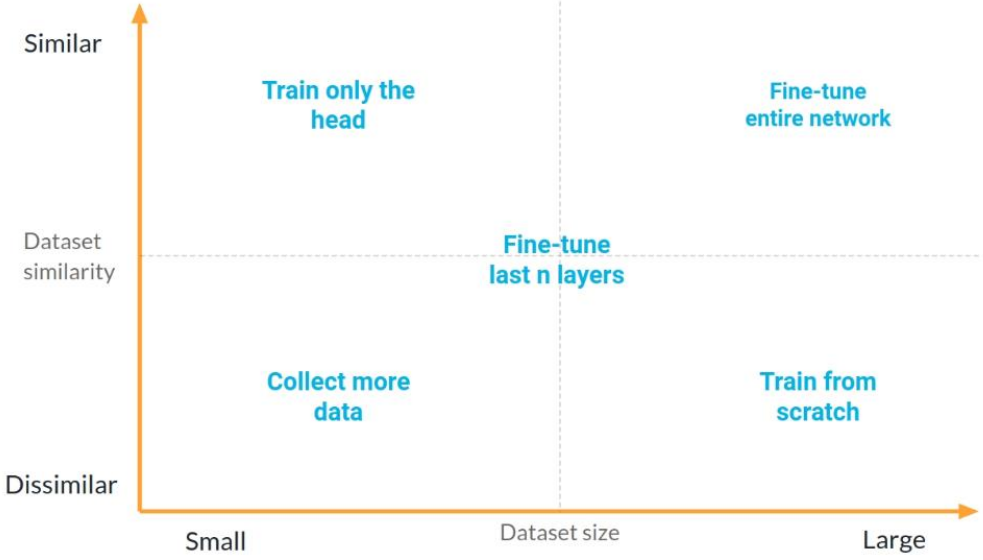


Your decision to use TL depends on:

- How similar your current dataset is to the used one in the main model.
- The size of your dataset.

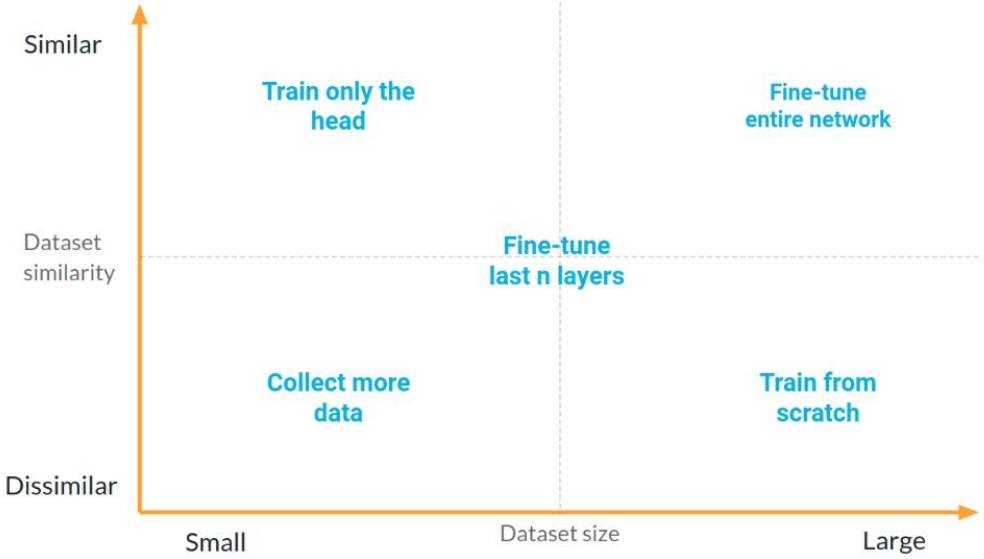
Techniques used

- Learning the head, keep earlier layers (Similar dataset, small dataset)
- Learning the whole model parameters or Fine Tune the model. (Huge dataset, non similar)



Fine Tuning

- Starting the model with its parameters and changing them while training.



Apply TL to Tiny ImageNet

Pretrained Models in PyTorch (Here)

Use Netron to display the Network Architecture graph.

What Do you think we can change to give us better result?

Break (10 minutes)

Satisfaction Survey

Project

If Consumed all Udacity Given GPU:

1- Email support to add extra hours to your account (scholarships-support@udacity.com)

2- You can use SageMaker studio Lab free GPU (4 hours / day)

3- Migrate to AWS account with SageMaker studio classic

If Consumed all Udacity Given GPU:

Common approach is to remove all versions of the libraries in the requirements.txt file and let pip decide which versions to install

AWS kernels aren't supporting python3.7 anymore.
If you are using Udacity's classroom instance, skip this part.

requirements (8) - Notepad

File Edit Format View Help

```
opencv-python-headless
cmake
lit
matplotlib
numpy
pillow
bokeh
torch
torchvision
tqdm
ipywidgets
livelossplot
pytest
pandas
seaborn
```

Remove packages' versions in requirements.txt.
Let pip install the versions compatible with one another and with Python 3.10

Pytest error in Udacity Workspace

```
In [4]: !pytest -vv src/data.py -k data_loaders
        #!python -m pytest -vv src/data.py -k data_loaders

        /bin/sh: 1: pytest: not found
```

Num_worker error

jupyter cnn_from_scratch Last Checkpoint: 04/15/2023 (unsaved changes)

File Edit View Insert Cell Kernel Help Not Trusted Python 3

Visualizing the output of your data loader is a great way to ensure that your data loading and preprocessing (including transforms such as rotations, translations, color transforms...) are working as expected.

In [5]:
```
%matplotlib inline
from src.data import visualize_one_batch, get_data_loaders


# use get_data_loaders to get the data_loaders dictionary. Use a batch_size
# of 5, a validation size of 0.01 and num_workers=-1 (all CPUs)
data_loaders = get_data_loaders(batch_size=5,valid_size=0.01, num_workers=0)

visualize_one_batch(data_loaders)
```

```
Reusing cached mean and std
Dataset mean: tensor([0.4638, 0.4725, 0.4687]), std: tensor([0.2699, 0.2706, 0.3018])
Reusing cached mean and std
```

In [10]:
```
import multiprocessing
n_workers = multiprocessing.cpu_count()
print(n_workers)
```

```
16
```

Menu Shrink • GPU -1 HR -6 MIN ENABLE

jupyter cnn_from_scratch Last Checkpoint: 04/15/2023 (unsaved changes)

File Edit View Insert Cell Kernel Help Not Trusted Python 3

Visualizing the output of your data loader is a great way to ensure that your data loading and preprocessing (including transforms such as rotations, translations, color transforms...) are working as expected.

In [4]:
```
%matplotlib inline
from src.data import visualize_one_batch, get_data_loaders


# use get_data_loaders to get the data_loaders dictionary. Use a batch_size
# of 5, a validation size of 0.01 and num_workers=-1 (all CPUs)
data_loaders = get_data_loaders(batch_size=5,valid_size=0.01, num_workers=-1)

visualize_one_batch(data_loaders)
```

```
    416         except InterruptedError:

/opt/conda/lib/python3.7/site-packages/torch/utils/data/_utils/signal_handling.py in handler(signum, frame)
     65         # Python can still get and update the process status successfully.
---> 66         _error_if_any_worker_fails()
     67         if previous_handler is not None:

RuntimeError: DataLoader worker (pid 3961) exited unexpectedly with exit code 1. Details are lost due to multiprocessing. Rer
unning with num_workers=0 may give better error trace.

The above exception was the direct cause of the following exception:

RuntimeError                              Traceback (most recent call last)
<ipython-input-4-e62dc8aa3d78> in <module>
      7 data_loaders = get_data_loaders(batch_size=5,valid_size=0.01, num_workers=-1)
      8
----> 9 visualize_one_batch(data_loaders)

/workspace/home/src/data.py in visualize_one_batch(data_loaders, max_n)
    148     # Then call the .next() method on the iterator you just
```

In [10]:
```
import multiprocessing
n_workers = multiprocessing.cpu_count()
```

Menu Shrink • GPU -1 HR -6 MIN ENABLE

Num_worker error

When setting **num_workers=-1**, the get_data_loaders function in data.py calculates number of cores for multi processing which seems to have a problem in pytorch old versions

```
In [10]: import multiprocessing
         n_workers = multiprocessing.cpu_count()
         print(n_workers)

         16
```

1.Use multi-threading instead: If **num_workers** continues to cause problems, consider using multi-threading (**num_workers=0** or **num_workers=1**), which might work better in certain environments.

Num_worker error

jupyter data.py a few seconds ago

File Edit View Language

```python

def get_data_loaders(
    batch_size: int = 32, valid_size: float = 0.2, num_workers: int = 0, limit: int = -1
):
    """
    Create and returns the train_one_epoch, validation and test data loaders.

    :param batch_size: size of the mini-batches
    :param valid_size: fraction of the dataset to use for validation. For example 0.2
                       means that 20% of the dataset will be used for validation
    :param num_workers: number of workers to use in the data loaders. Use -1 to mean
                        "use all my cores"
    :param limit: maximum number of data points to consider
    :return a dictionary with 3 keys: 'train_one_epoch', 'valid' and 'test' containing respectively the
            train_one_epoch, validation and test data loaders
    """

    if num_workers == -1:
        # Use all cores
        num_workers = multiprocessing.cpu_count()

    # We will fill this up later
    data_loaders = {"train": None, "valid": None, "test": None}

    base_path = Path(get_data_location())

    # Compute mean and std of the dataset
    mean, std = compute_mean_and_std()
    print(f"Dataset mean: {mean}, std: {std}")

```

Any change you do to any .py file, restart your notebook to see an action

Access the checkpoints folder

Solved

App.ipynb Problems

List out of index error

```python
from ipywidgets import VBox, Button, FileUpload, Output, Label
from PIL import Image
from IPython.display import display
import io
import numpy as np
import torchvision
import torchvision.transforms as T
import torch

learn_inf = torch.jit.load("checkpoints access/transfer_exported.pt")

# Load image that has been uploaded
img = Image.open("eiffel-tower.jpg") #eiffel-tower.jpg #Dead sea.jpg

img.load()

ratio = img.size[0] / img.size[1]
c = img.copy()
c.thumbnail([ratio * 200, 200])
display(c)

    # Transform to tensor
timg = T.ToTensor()(img).unsqueeze_(0)

    # Calling the model
softmax = learn_inf(timg).data.cpu().numpy().squeeze()

    # Get the indexes of the classes ordered by softmax
    # (larger first)
idxs = np.argsort(softmax)[::-1]

# Loop over the classes with the largest softmax
for i in range(5):
    # Get softmax value
    p = softmax[idxs[i]]
    # Get class name
    landmark_name = learn_inf.class_names[idxs[i]]
    print(f"{landmark_name} (prob: {p:.2f})")
```

```
16.Eiffel_Tower (prob: 0.50)
14.Terminal_Tower (prob: 0.20)
47.Prague_Astronomical_Clock (prob: 0.08)
48.Whitby_Abbey (prob: 0.07)
19.Vienna_City_Hall (prob: 0.03)
```

Cell Output

Cell Input

- If you faced this error, One way to work around is to use an [online tool](#) that would convert the ipynb file to HTML file.

```
In [2]: !python src/create_submit_pkg.py
    return inline.renderer.finalize(data)
  File "/home/ec2-user/anaconda3/envs/amazonei_pytorch_latest_p37/lib/python3.7/site-packages/mistune/renderers.py", line 22
0, in finalize
    return ''.join(data)
  File "/home/ec2-user/anaconda3/envs/amazonei_pytorch_latest_p37/lib/python3.7/site-packages/mistune/block_parser.py", line
291, in _iter_render
    yield method(children, *params)
  File "/home/ec2-user/anaconda3/envs/amazonei_pytorch_latest_p37/lib/python3.7/site-packages/nbconvert/filters/markdown_mist
une.py", line 181, in block_code
    lang = info.strip().split(None, 1)[0]
IndexError: list index out of range
Traceback (most recent call last):
  File "src/create_submit_pkg.py", line 40, in <module>
    create_submit_pkg()
  File "src/create_submit_pkg.py", line 20, in create_submit_pkg
    subprocess.check_call(cmd_line, shell=True)
  File "/home/ec2-user/anaconda3/envs/amazonei_pytorch_latest_p37/lib/python3.7/subprocess.py", line 363, in check_call
    raise CalledProcessError(retcode, cmd)
subprocess.CalledProcessError: Command 'jupyter nbconvert --to html cnn_from_scratch.ipynb' returned non-zero exit status 1.
```

Any Question?

Thank you

UDACITY