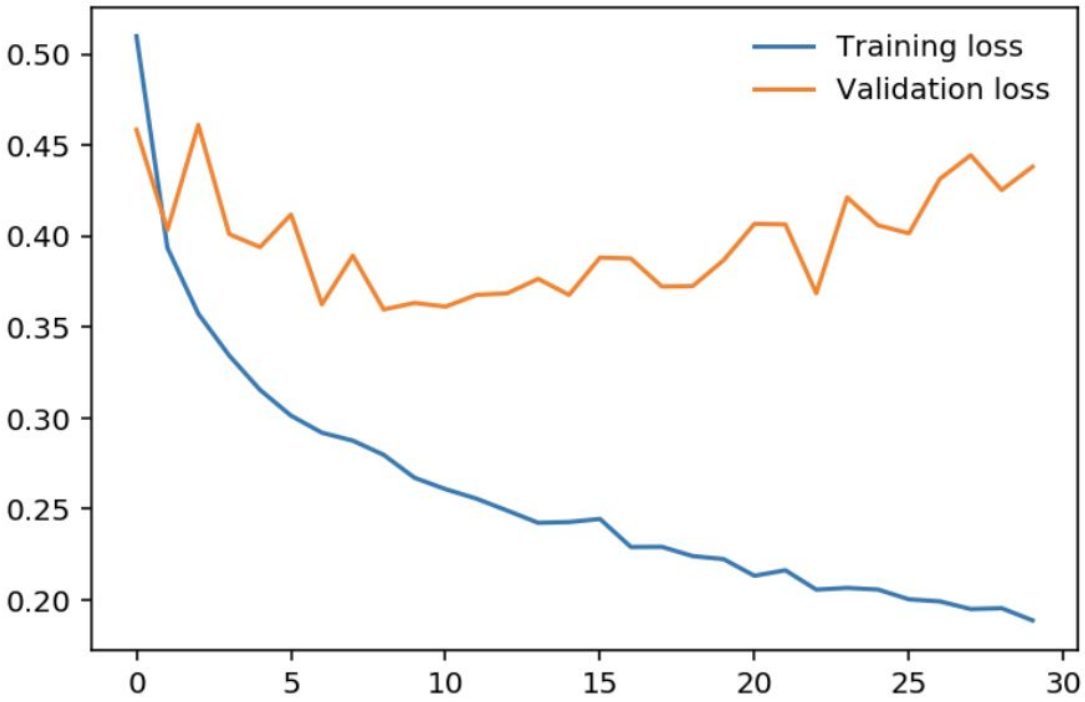
Reference: Lesson 5: Introduction of PyTorch: 16, 17, 18, 19

- # Dropout
    - For definition, problem statement and solution refer to https://docs.google.com/document/d/1S1sFxUBygvdNoWRDX5YSQzzzvNEmrFvmKoYbcsmw9G8/edit



    - **Problem**: graph above show overfitting happening

More about overfitting and underfitting, go to lesson 3: https://docs.google.com/document/d/1S1sFxUBygvdNoWRDX5YSQzzzvNEmrFvmKoYbcsmw9G8/edit

- **Solution**: the ultimate goal of any deep learning model is to make predictions on new data, so we should strive to get the lowest validation loss possible
    - Early-stopping
    - Dropout

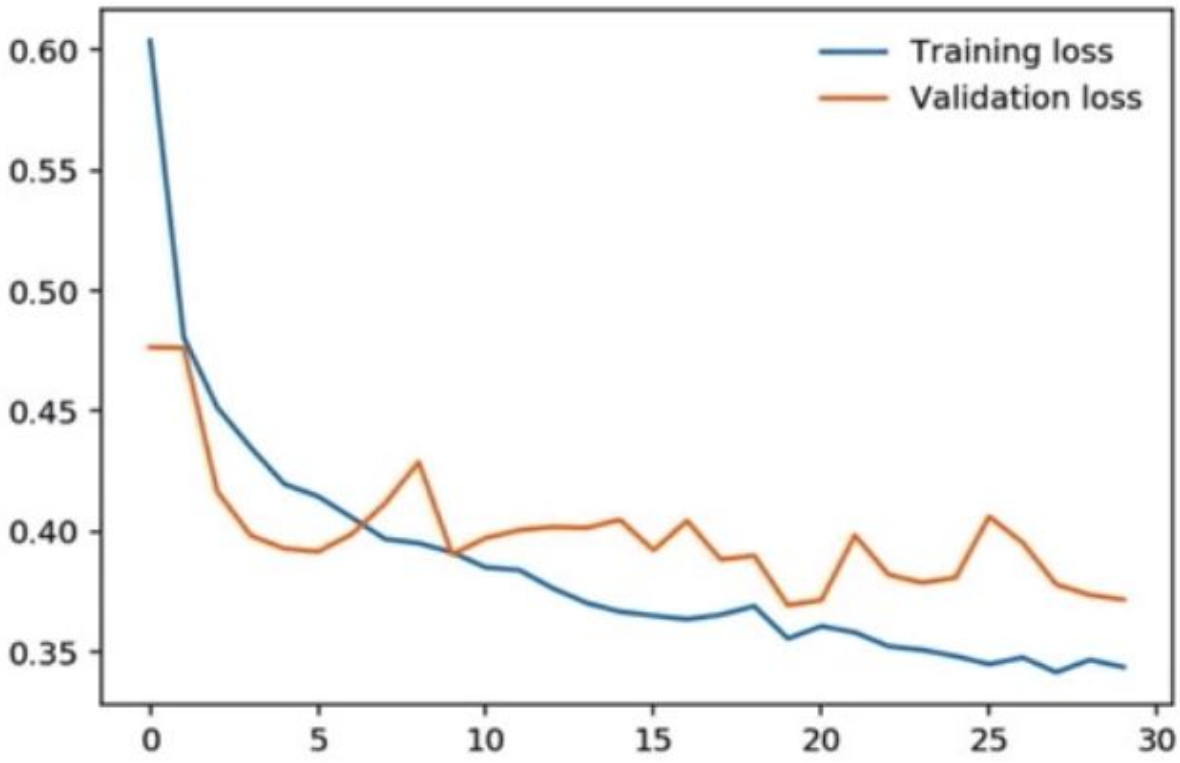
- **How do we perform drop out with PyTorch?**
    - **Rule**:
        - During training we want to use dropout to prevent overfitting
        - But during inference we want to use the entire network. So, we need to turn off dropout during validation, testing, and whenever we're using the network to make predictions/inference.

```
# turn off gradients
with torch.no_grad():

    # set model to evaluation mode
    model.eval()

    # validation pass here
    for images, labels in testloader:
        ...

# set model back to train mode
model.train()
```

- `model.eval()` This will set the model to evaluation mode where the dropout probability is 0
- `model.train()` This will set the model back to training mode where the dropout equal to the value we set
  Example how to set the dropout value:

```
# Dropout module with 0.2 drop probability
        self.dropout = nn.Dropout(p=0.2)
```

- Result after applying dropout (please refer to the code in Notebook)



- Validation loss stick closer to training loss as we train → reduce overfitting.
- If we keep training, validation loss tends to drop down more

- **Inference**
    - Inference is the stage in which a trained model is used to infer/predict the testing samples and comprises of a similar forward pass as training to predict the values. https://blog.exxactcorp.com/discover-difference-deep-learning-training-inference/
    - As mentioned in rule of applying drop out, we need to turn off the drop during this stage

==========

Mini Summary:

- Why need to apply dropout
- How to perform dropout with PyTorch
- Rule of applying PyTorch

==========

- **How to save and load model with PyTorch**
    - Why is it important ?
        - You want to make prediction later?
        - You want to train with new data
    - **Setup (load data, create model architecture and training)**
        - In the notebook, we import the needed libraries
        - What is fc_model ? this is just a custom class written by the tutor for convenience, it contains model architecture creation and training code, here is the full source code: https://github.com/udacity/deep-learning-v2-pytorch/blob/master/intro-to-pytorch/fc_model.py
    - PyTorch
        - The parameters for PyTorch networks are stored in a model's **state_dict;** it contains weight and bias metric of each layer
        - Good reference if new with Python dictionary: https://www.w3schools.com/python/python_dictionaries.asp

```
print("The state dict keys: \n\n", model.state_dict().keys())

#save the state_dict() to file checkpoint.pth
torch.save(model.state_dict(), 'checkpoint.pth')

#load file checkpoint.pth contains parameters of PyTorch network
state_dict = torch.load('checkpoint.pth')

#load state_dict to network directly
model.load_state_dict(state_dict)
```

- **Issue:** error throw when try to load state_dict to model with different architecture
  Example:
    - In the notebook, you create a model_1

```
model_1 = fc_model.Network(784, 10, [512, 256, 128])
```

    - You save it to a checkpoint_1; now if we create another model, model_2, with different architecture; it will throw error.

```
# Try this
model_2 = fc_model.Network(784, 10, [400, 200, 100])
# This will throw an error because the tensor sizes are wrong!
model.load_state_dict(state_dict)
```

- **Rule**: Loading the state dict works only if the model architecture is exactly the same as the checkpoint architecture.

- **How to resolve issue above?**
    - Rebuild the model exactly as it was when trained. Information about the model architecture needs to be saved in the checkpoint, along with the state dict.

==========

Mini Summary

- Why save and load model is important?
- How to do that with PyTorch
- What is the issue while trying to load the model?
- How to resolve that issue?

- # Loading Image Data
    - Learn how to load real image not the artificial dataset
    - The notebook's goal is to create a neural network to distinguish cat and dog

```
dataset = datasets.ImageFolder('path/to/data', transform=transform)
```

- ‘path/to/data’ is path to directory folder contains images
- transform is processing steps built with transform module from torchvision; normally this used in pipeline that help the codee more clean and organized.

    Example:

    In PyTorch, we use compose to build a pipeline. Below is pipeline to scale, then crop then convert to tensor

```
transform = transforms.Compose([transforms.Resize(255),
                                transforms.CenterCrop(224),
                                transforms.ToTensor()])
```

- **What is Data Loader**
    - It takes a dataset and returns batches of images and the corresponding labels
    - It was passed by the data loaded from ImageFolder
- **What is Data Augmentation**
    - A common strategy for training neural networks is to introduce randomness in the input data itself.
    - This will help your network generalize as it's seeing the same images but in different locations, with different sizes, in different orientations

- Example: To randomly rotate, scale and crop, then flip your images you would define your transforms like this:

```
train_transforms = transforms.Compose([transforms.RandomRotation(30),
                                       transforms.RandomResizedCrop(224),
                                       transforms.RandomHorizontalFlip(),
                                       transforms.ToTensor(),
                                       transforms.Normalize([0.5, 0.5, 0.5],
[0.5, 0.5, 0.5])])
```

==========

Mini Summary

- How to load real image data with PyTorch ?
- How to use pipeline in PyTorch to pre-process images