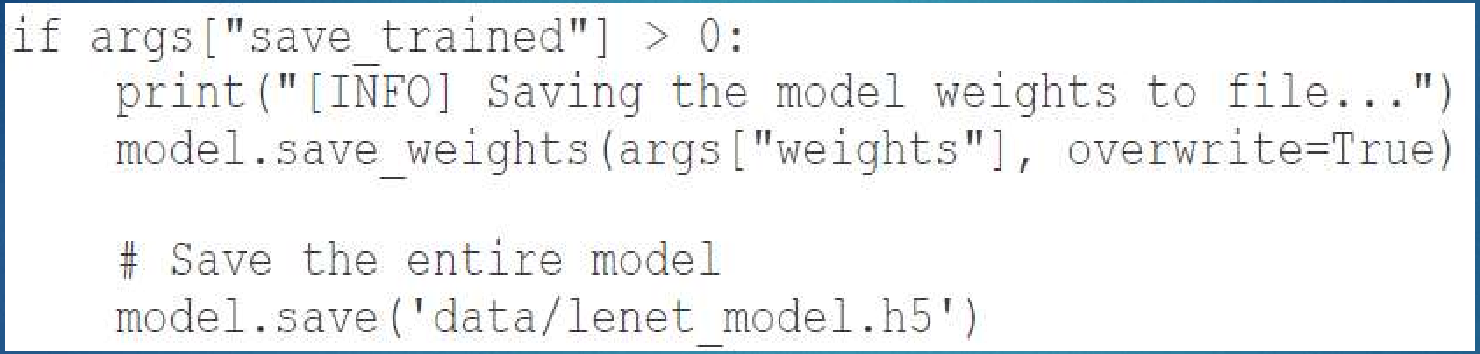
Building a deep learning application

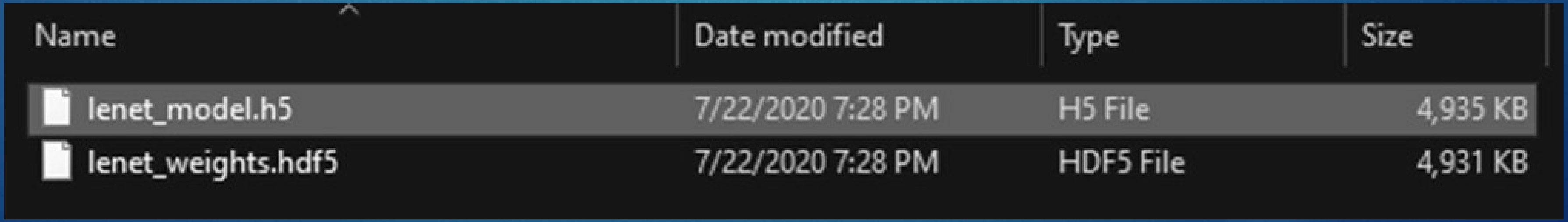
DR FAKHRELDIN SAEED

Save and Load the Models (Keras)

- *model.save_weights()*
- Keras provides another save option: *model.save()*.



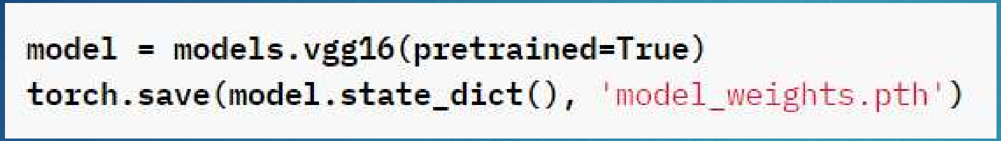
```
if args["save_trained"] > 0:
    print("[INFO] Saving the model weights to file...")
    model.save_weights(args["weights"], overwrite=True)

    # Save the entire model
    model.save('data/lenet_model.h5')
```



```
Name                          Date modified          Type          Size
lenet_model.h5                7/22/2020 7:28 PM      H5 File       4,935 KB
lenet_weights.hdf5            7/22/2020 7:28 PM      HDF5 File     4,931 KB
```

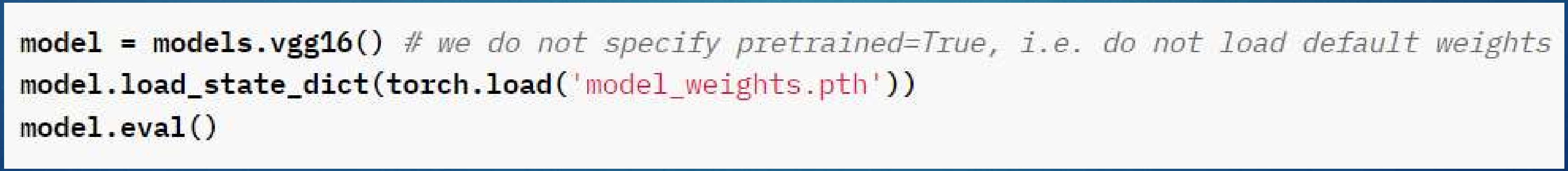
Save and Load the Models(Pytorch)

- PyTorch models store the learned parameters in an internal state dictionary, called state_dict.
- These can be persisted via the **torch.save** method:

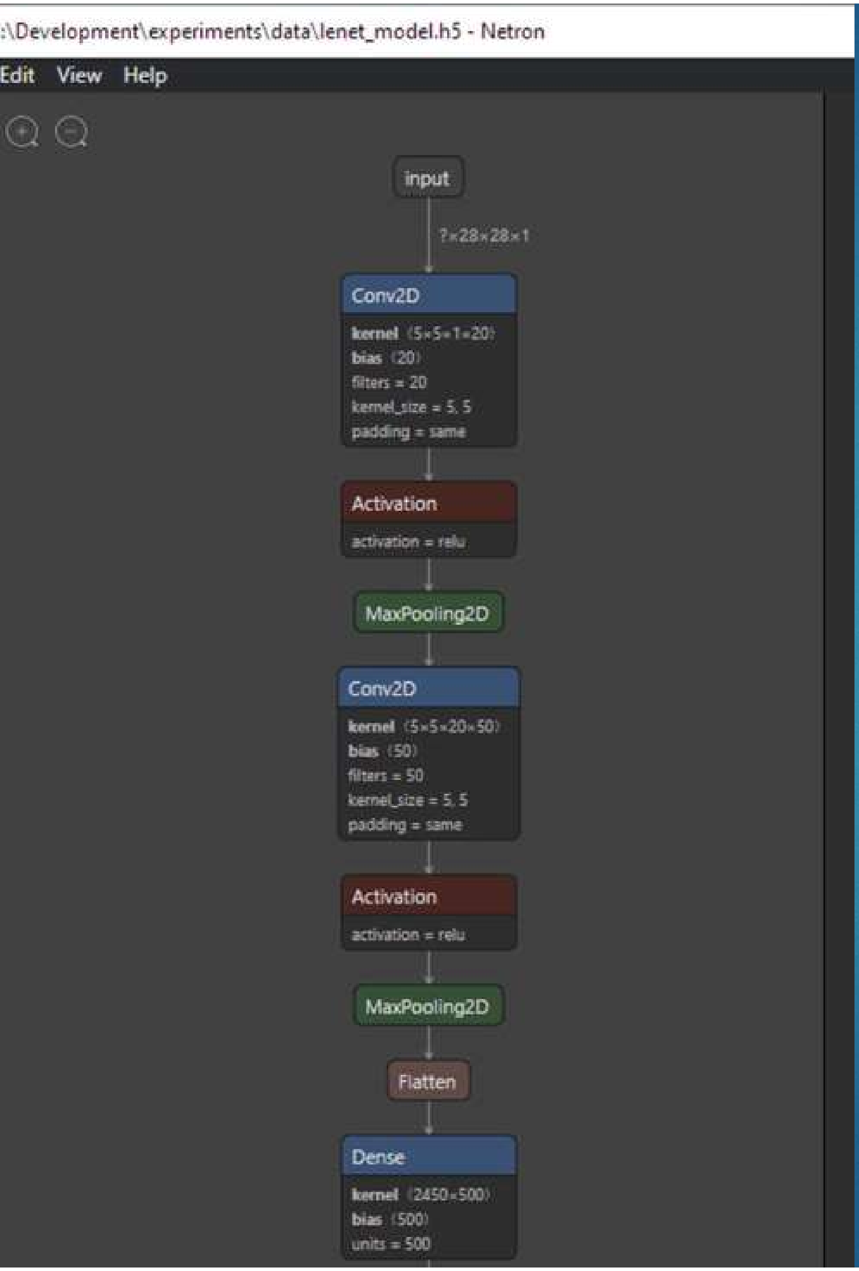


```
model = models.vgg16(pretrained=True)
torch.save(model.state_dict(), 'model_weights.pth')
```

- To load model weights, you need to create an instance of the same model first, and then load the parameters using load_state_dict() method.



```
model = models.vgg16() # we do not specify pretrained=True, i.e. do not load default weights
model.load_state_dict(torch.load('model_weights.pth'))
model.eval()
```

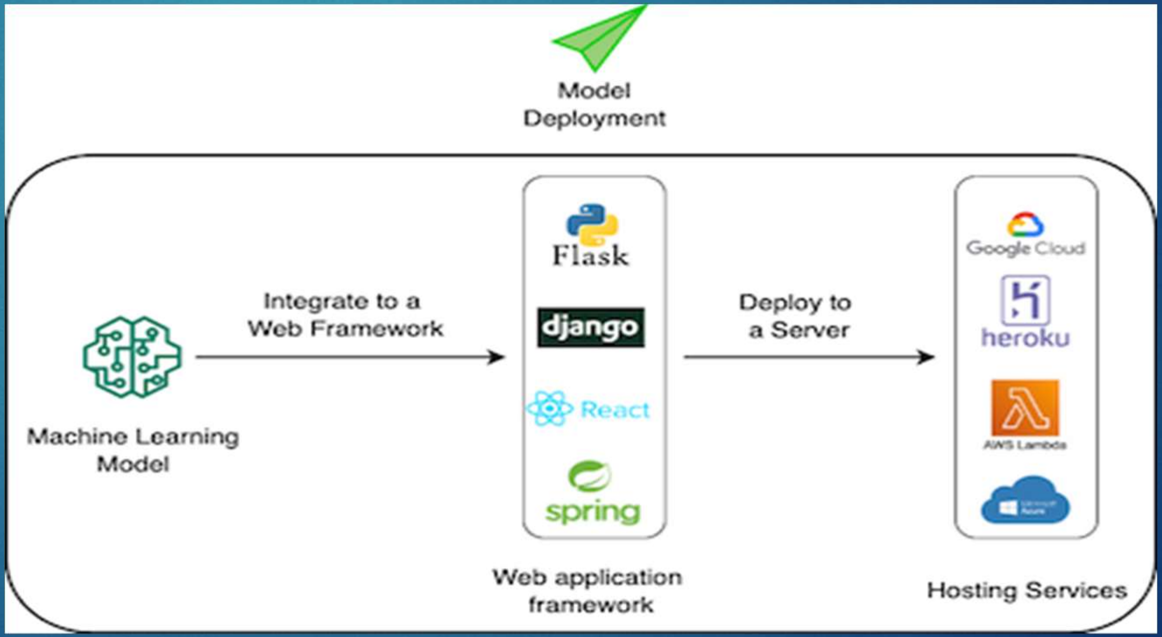


Visualizing Models

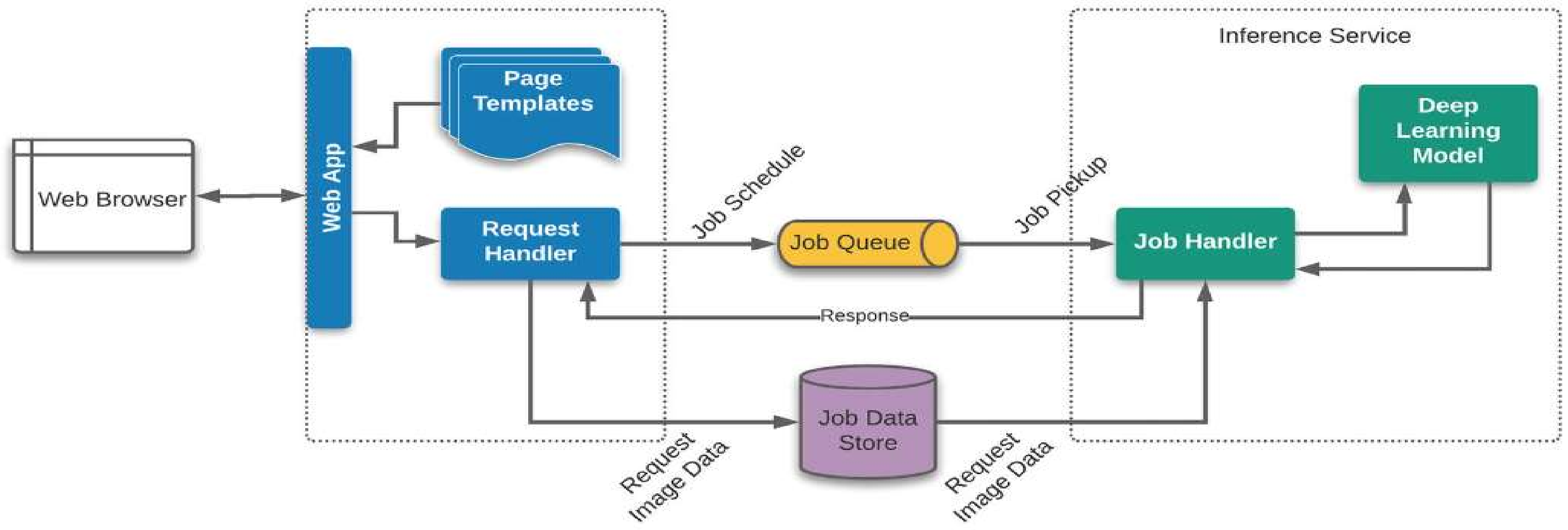
- When building a deep learning model, it is often better to be able to visualize the model.
- Visualize Model Structures:
  - We can use the **plot_model** function from the **tf.keras.utils** package to plot the structure of a Keras or tf.keras model.
  - Netron is an opensource visualizer for neural network, deep learning, and machine learning models. It is available through its GitHub page(*https://github.com/lutzroeder/netron*)

Deploying the Model as a Web Application

- We can use a framework to turn our model into a web application.
  - **Flask**
  - Django
  - Bottle
  - Web2Py



https://towardsdatascience.com/deploying-machine-learning-models-with-heroku-4dec1df87f71

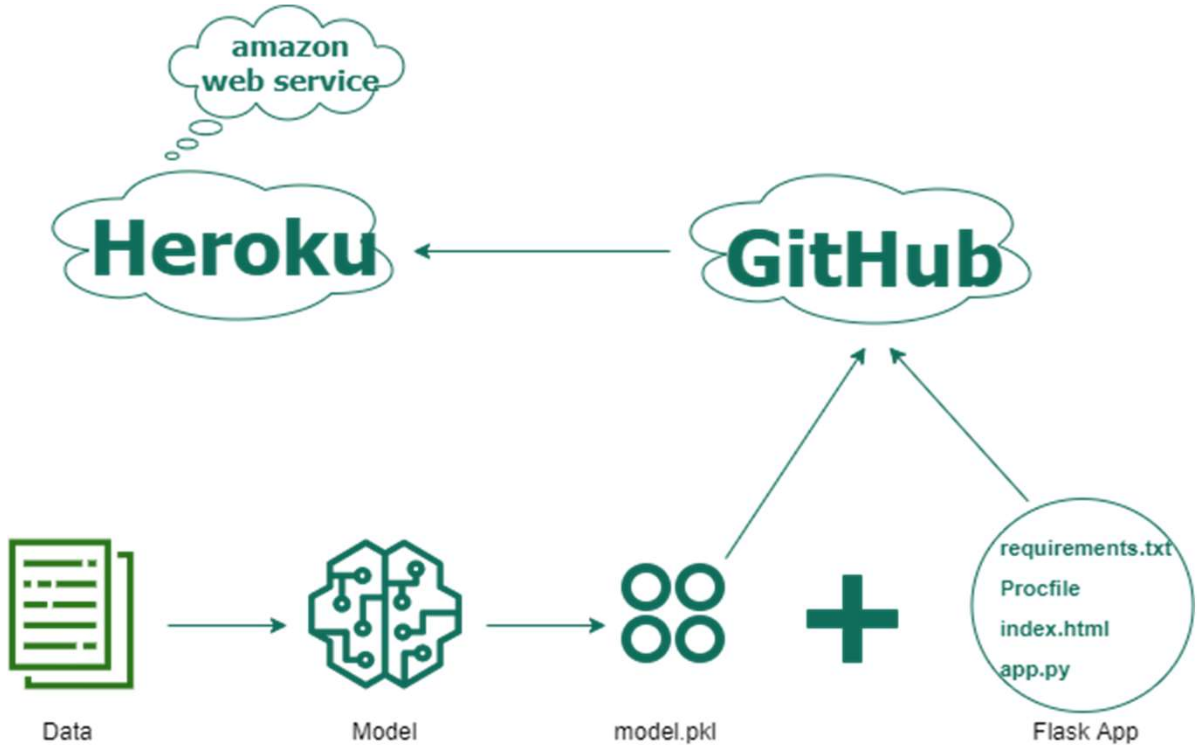


Scaling Up Your Web Application

Publish your application

- Build MVC web applications and lightweight APIs using **Flask**.
- Easily deploy, manage, and scale your apps by following simple procedures in **Heroku**.
- Deploy apps from Git, CI systems, or **GitHub**.
- Heroku runs all the applications in a dynamic, secure, and smart container.
- Manage applications from a detailed dashboard or by using a CLI.

Publish your application



https://towardsdatascience.com/machine-learning-model-deployment-on-heroku-using-flask-467acb4a34da

Steps for deployment on Heroku using Flask

1. Create ML Model and save (pickle) it
2. Create Flask files for UI and python main file (app.py) that can unpickle the machine learning model from step 1 and do predictions.
3. Create **requirements.txt** to setup Flask web app with all python dependencies
4. Create Procfile to initiate Flask app command
5. Commit files from Step 1, 2, 3 & 4 in the Github
6. Create account/Login on Heroku, create an app, connect with Github, and select branch
7. Select manual deploy (or enable Automatic deploys) on Heroku

Reading

- 9. Deploying Your Model as a Web Application | Deep Learning on Windows: Building Deep Learning Computer Vision Systems on Microsoft Windows (oreilly.com)

- Lab: https://towardsdatascience.com/complete-guide-on-model-deployment-with-flask-and-heroku-98c87554a6b9
- https://towardsdatascience.com/machine-learning-model-deployment-on-heroku-using-flask-467acb4a34da