SDSC HPC-DS Summer Institute 2025
Deep Learning - Transfer Learning

SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

Deep Learning
Transfer Learning

Mai H. Nguyen, Ph.D.

DEEP LEARNING OVERVIEW

- **Neural Network Basics**
  - Processing Unit
  - Activation Function
  - Loss Function
- **Deep Learning Fundamentals**
  - Deep Network Layers
  - DL Architectures
  - DL Libraries
- **Transfer Learning**
  - Transfer Learning Concepts
  - Transfer Learning Demo

Transfer Learning

- **To overcome challenges of training model from scratch:**
  - Insufficient data
  - Very long training time
- **Use pre-trained model**
  - Trained on another dataset
  - This serves as starting point for model
  - Then train model on current dataset for current task

Transfer Learning Approaches

- **Feature extraction**
  - Remove classification layer from pre-trained model
  - Treat rest of network as feature extractor
  - Use features to train new classifier
    - “top model” or “classification head”
- **Fine tuning**
  - Tune weights in some layers of original model (along with weights of top model)
  - Train model for current task using new dataset

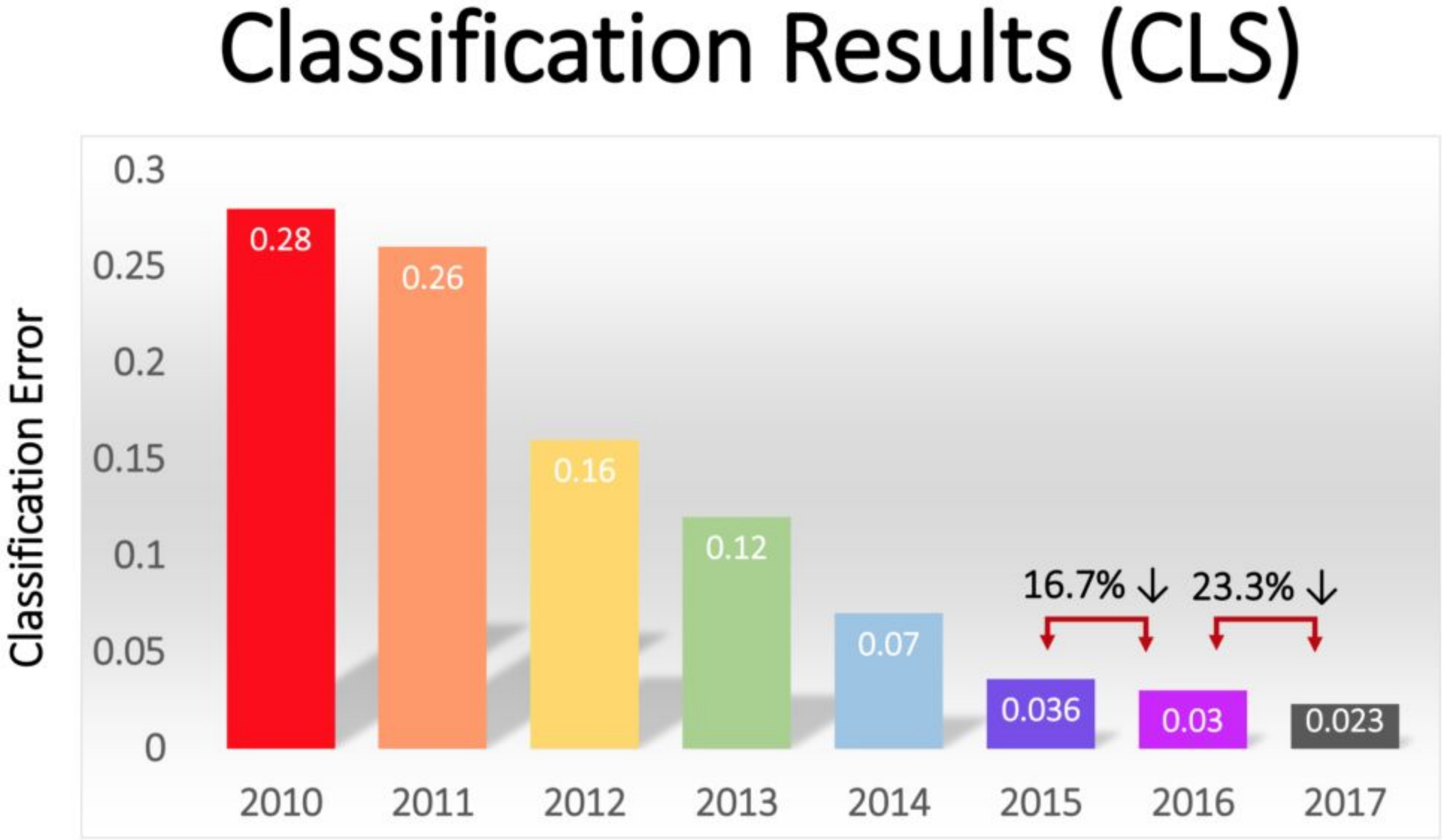
CNNs for Transfer Learning

- **Popular architectures**
  - AlexNet
  - GoogLeNet
  - VGGNet
  - ResNet
- **All winners of ILSVRC**
  - ImageNet Large Scale Visual Recognition Challenge
  - Annual competition on vision tasks on ImageNet data

ImageNet

- **Database**
  - Developed for computer vision research
  - ~14,000,000 images hand-annotated
  - ~22,000 categories
- **ILSVRC History**
  - Started in 2010
  - Image classification task: 1,000 object categories
  - Image classification error rate
    - 2010: 28.20%   (conventional image processing techniques)
    - 2012: 15.30% (AlexNet)
    - 2015:   3.57% (ResNet; better than human performance)
    - 2016:   2.99% (16.7% error reduction)
    - 2017:   2.25% (23.3% error reduction)

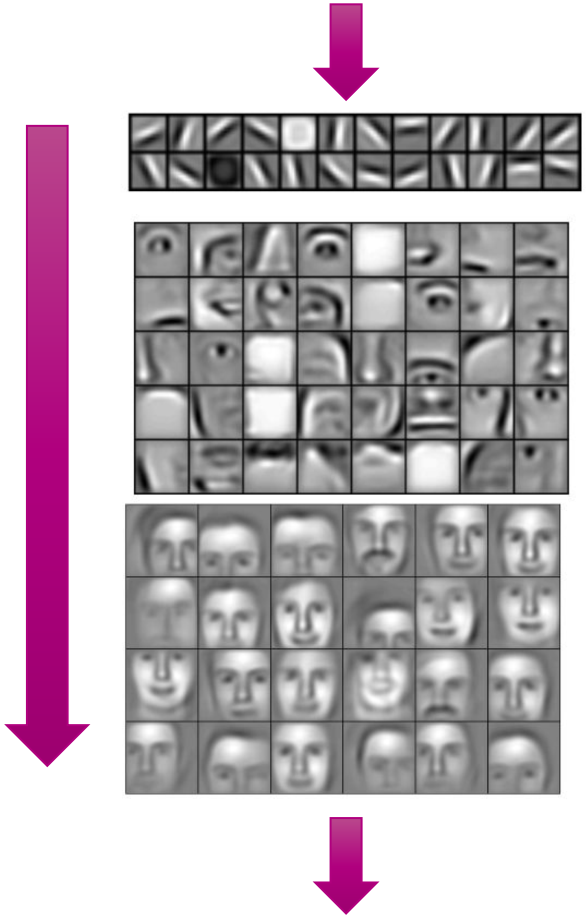
Results on ImageNet Classification



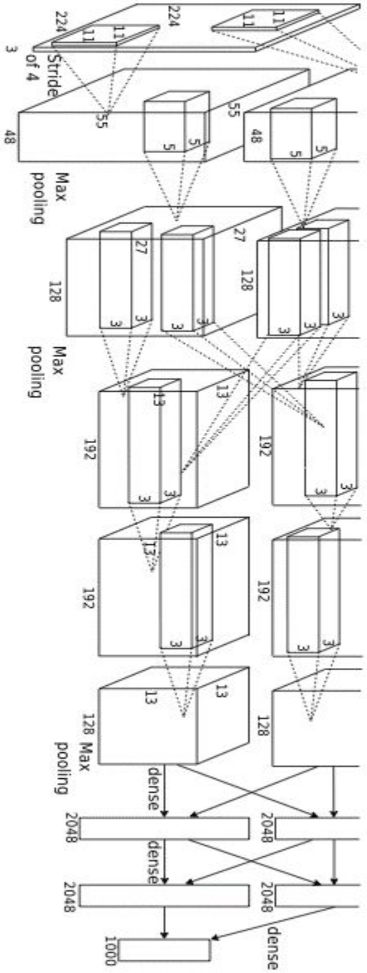
Transfer Learning

Input

Learned hierarchy

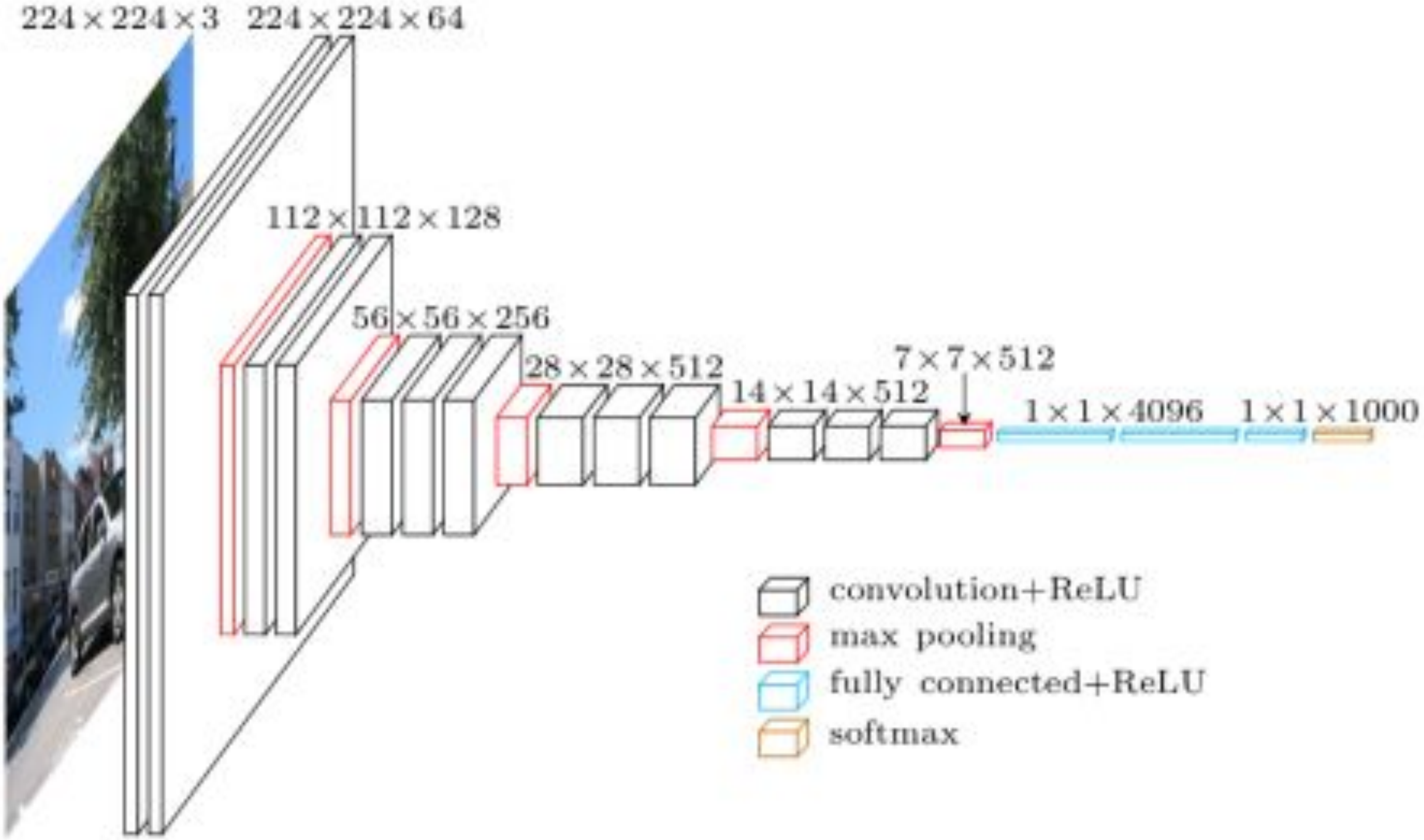


Output



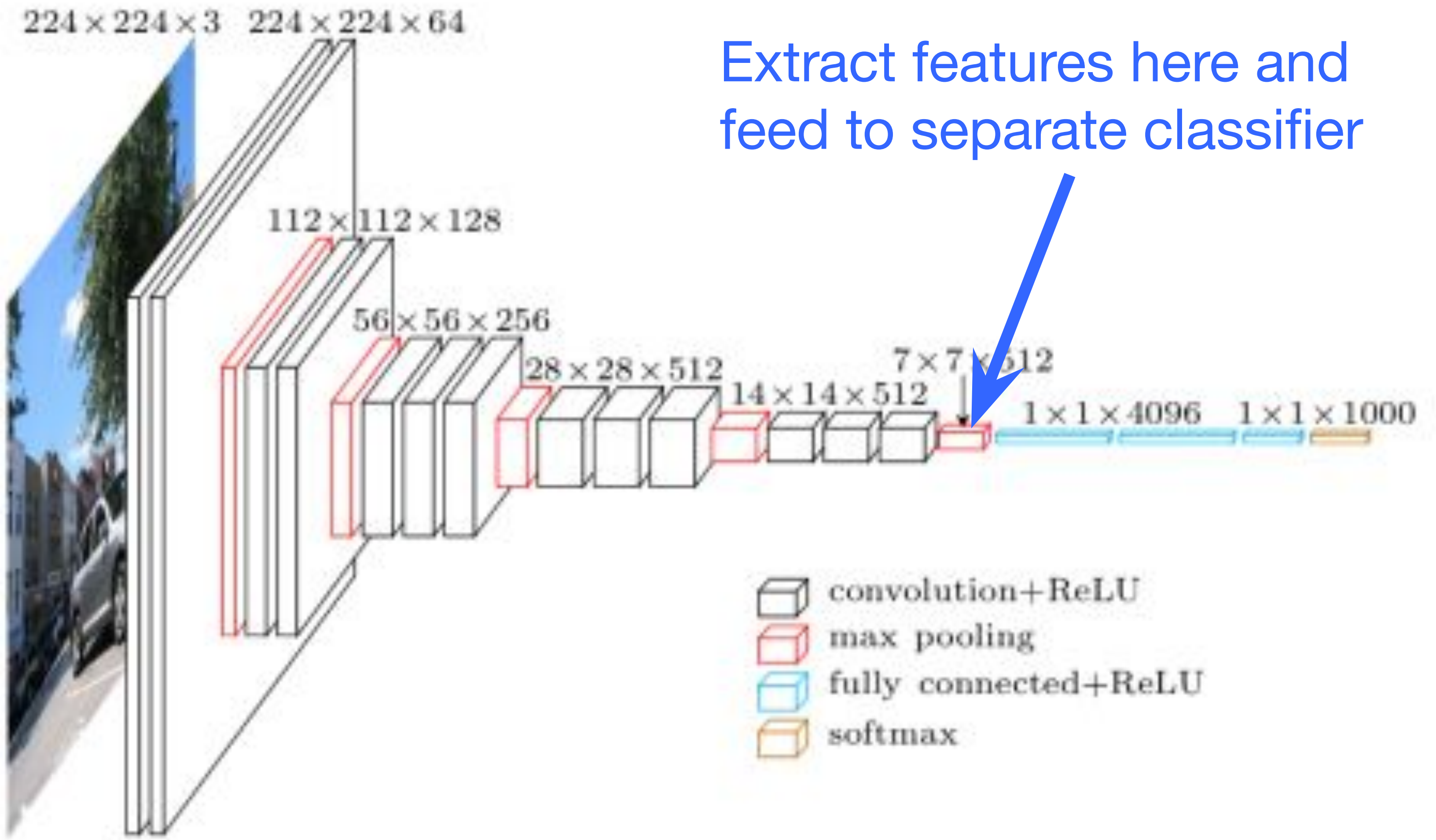
Lee et al. ‘Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations’ ICML 2009

Pre-Trained Model



https://www.cs.toronto.edu/~frossard/post/vgg16/

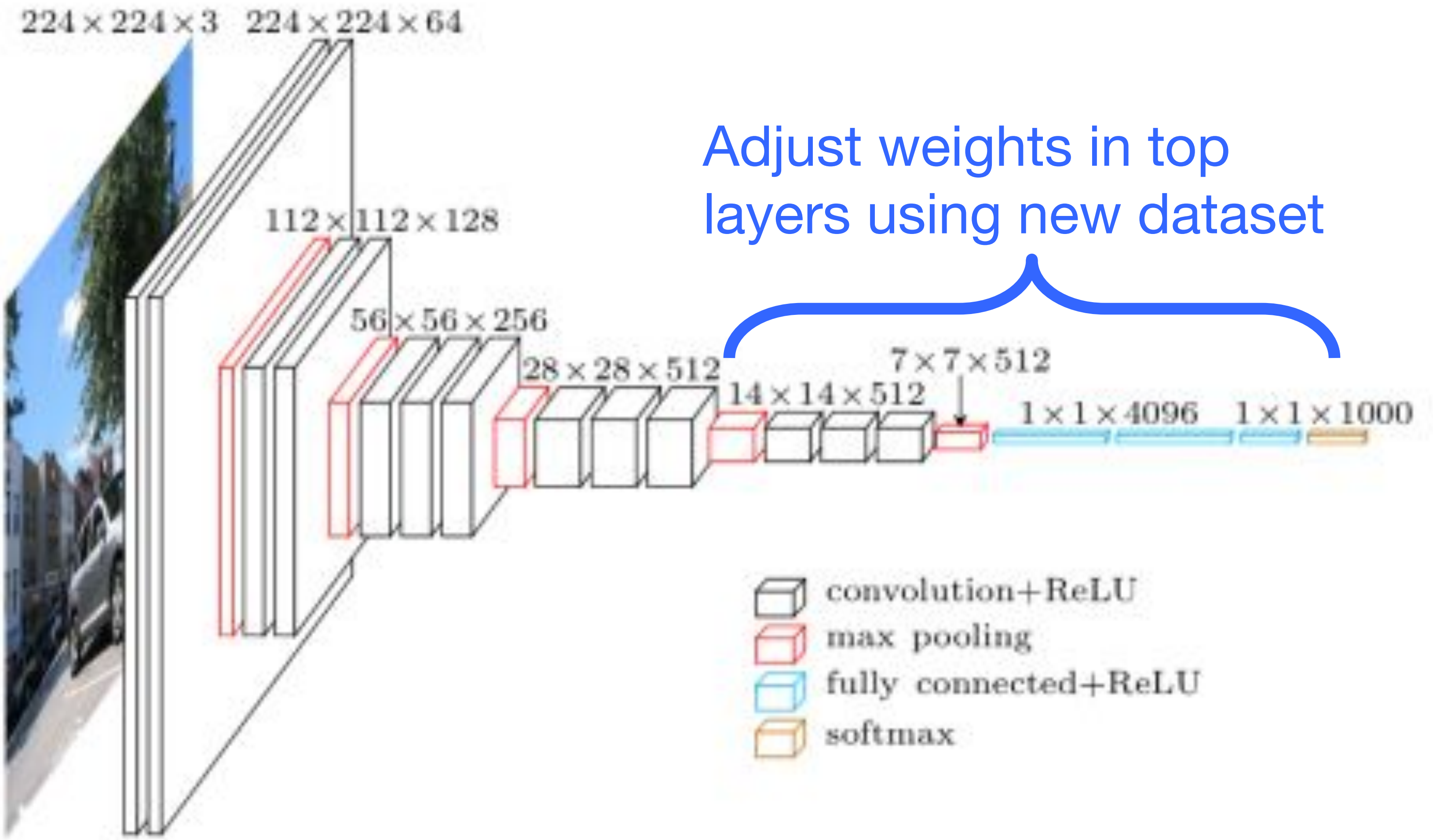
Transfer Learning - Feature Extraction



Extract features here and feed to separate classifier

https://www.cs.toronto.edu/~frossard/post/vgg16/

Transfer Learning - Fine Tuning



Adjust weights in top layers using new dataset

https://www.cs.toronto.edu/~frossard/post/vgg16/

Practical Tips for Transfer Learning

- **Learning rate**
  - Use very small learning rate for fine tuning. Don’t want to destroy what was already learned.
- **Start with properly trained weights**
  - Train top-level classifier first, then fine tune lower layers.
  - Top model with random weights may have negative effects on when fine tuning weights in pre-trained model
- **Data augmentation**
  - Simple ways to slightly alter images
    - Horizontal/vertical flips, random crops, translations, rotations, etc.
  - Use to artificially expand your dataset

Transfer Learning Hands-On

- **Data**
  - Cats and dogs images from Kaggle

- **Exercises**
  - Feature extraction
    - Use pre-trained CNN to extract features from images
    - Train neural network to classify cats/dogs using extracted features
  - Fine tune
    - Adjust weights of last few layers of pre-trained CNN and top classifier model through training

Data

- **Subset of Dogs Vs. Cats dataset from Kaggle**
  - https://www.kaggle.com/c/dogs-vs-cats
- **Train**
  - 1000 cats + 1000 dogs
- **Validation**
  - 200 cats + 200 dogs
- **Test**
  - 200 cats + 200 dogs

Setup

- **Login to Expanse**
  - Open terminal window on local machine
  - `ssh login.expanse.sdsc.edu -l <account>`
- **Pull latest from repo**
  - `git pull`
  - URL: https://github.com/sdsc/sdsc-summer-institute-2025.git

Server Setup for PyTorch Lightning- Command Line

- **In terminal window**
  - jupyter-gpu-shared-ptl
    - Alias for:

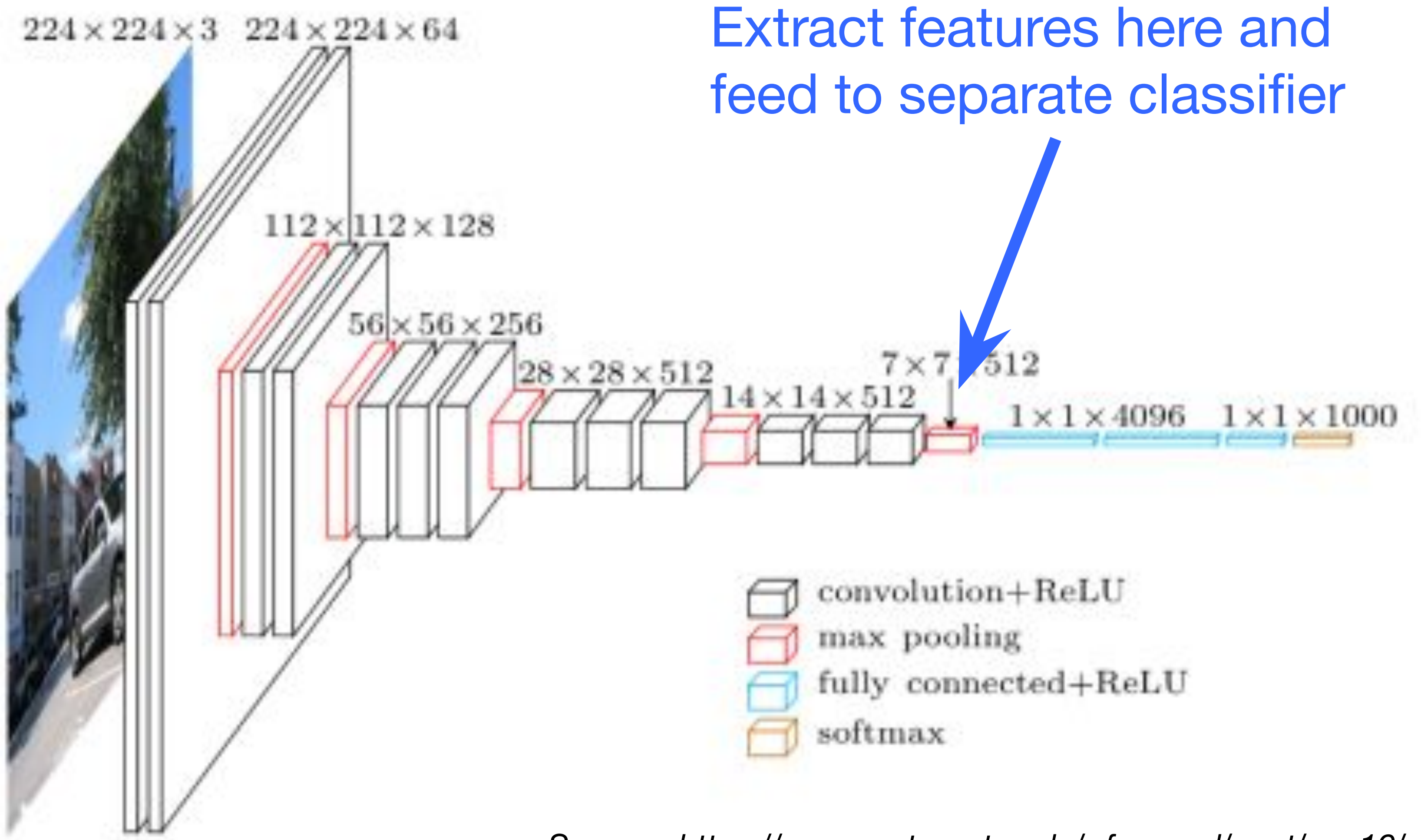
    -
      ```
      galyleo launch --account ${SI25_ACCOUNT} --reservation ${SI25_RES_GPU} --partition gpu-shared --qos ${SI25_QOS_GPU} --cpus 10 --memory 92 --gpus 1 --time-limit 04:00:00 --conda-yml ptl.yaml --mamba --cache
      ```
- **To check queue**

  -
    ```
    squeue -u $USER
    ```

Data

- **In terminal window in Jupyter Lab, do the following**
- **Get counts of images**
  - ls // Should see data
  - ls –l data/catsVsDogs/train/cats/* | wc -l
  - ls –l data/catsVsDogs/train/dogs/* | wc -l
  - ls –l data/catsVsDogs/val/cats/* | wc -l
  - ls –l data/catsVsDogs/val/dogs/* | wc -l
  - ls –l data/catsVsDogs/test/cats/* | wc -l
  - ls –l data/catsVsDogs/test/dogs/* | wc -l

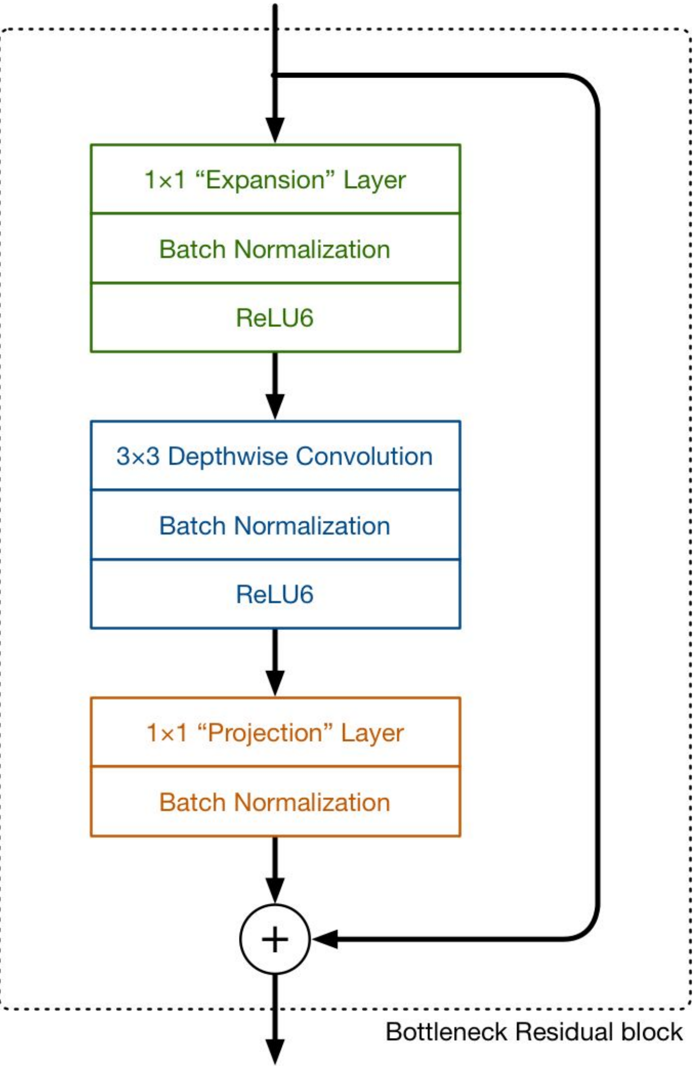
TRANSFER LEARNING - FEATURE EXTRACTION



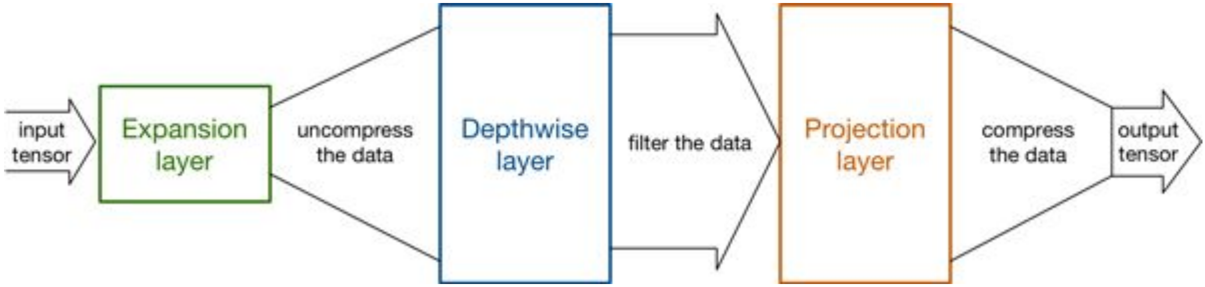
Extract features here and feed to separate classifier

Source: https://www.cs.toronto.edu/~frossard/post/vgg16/

MobileNetV2



- CNN
- Lightweight architecture
- Designed for mobile devices

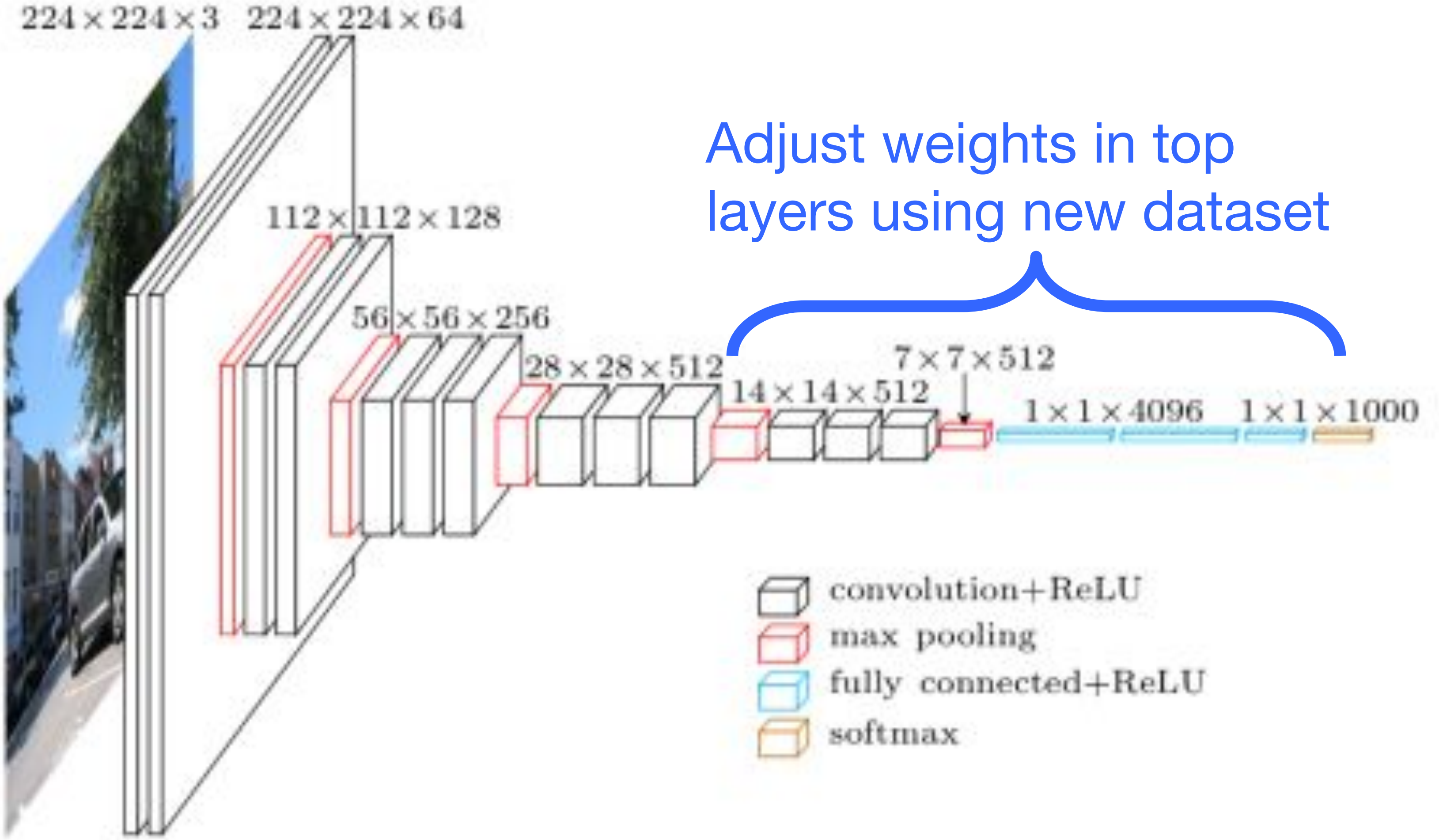


https://machinethink.net/blog/mobilenet-v2/

Feature Extraction Overview

- **Code**
  - feature_extraction_ptl.ipynb
- **Data**
  - Set image dimensions & location
  - Read images from folder in batches
- **Model**
  - Load model pre-trained on ImageNet data
  - Freeze weights in pre-trained model to use as feature extractor
  - Add top model to classify cats vs dogs
  - Model = pre-trained base model + top model classifier
- **Train model**
  - Use training data to adjust top model weights
- **Evaluate model**
  - Calculate accuracy, etc.
  - Perform inference on test images

TRANSFER LEARNING - FINE TUNING



Adjust weights in top layers using new dataset

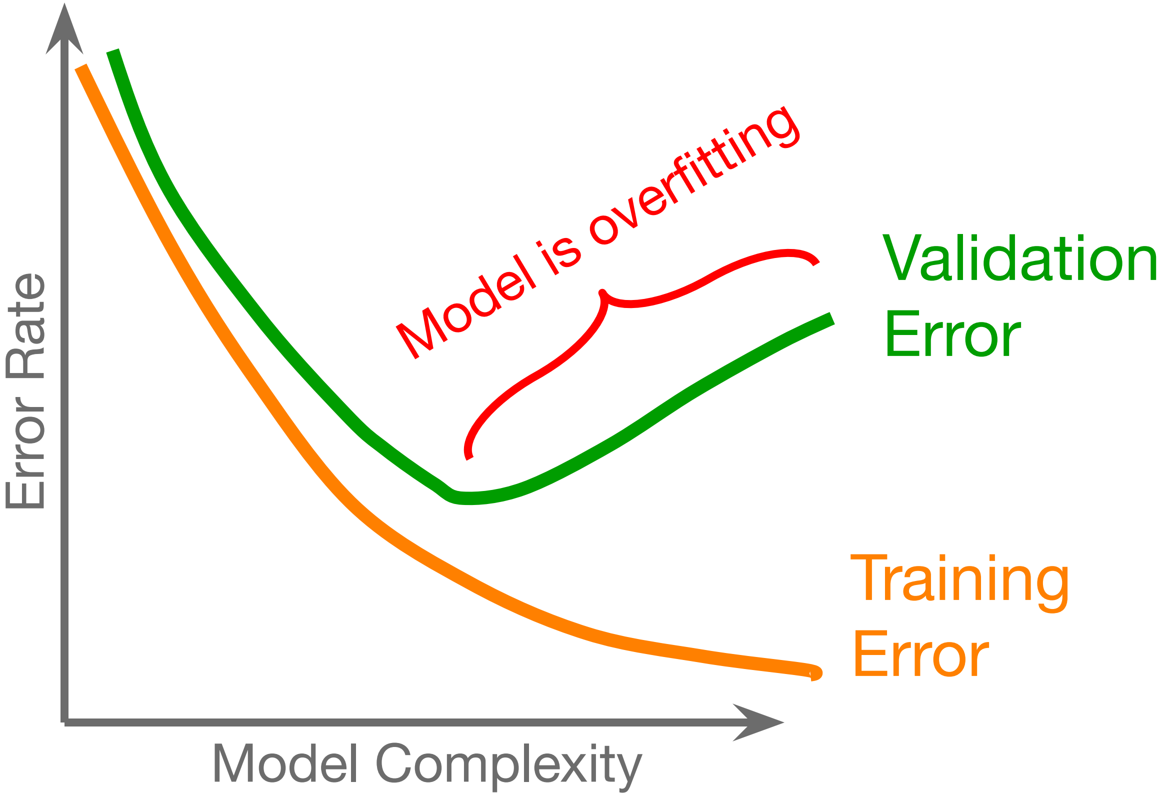
Source: https://www.cs.toronto.edu/~frossard/post/vgg16/

Fine Tune Overview

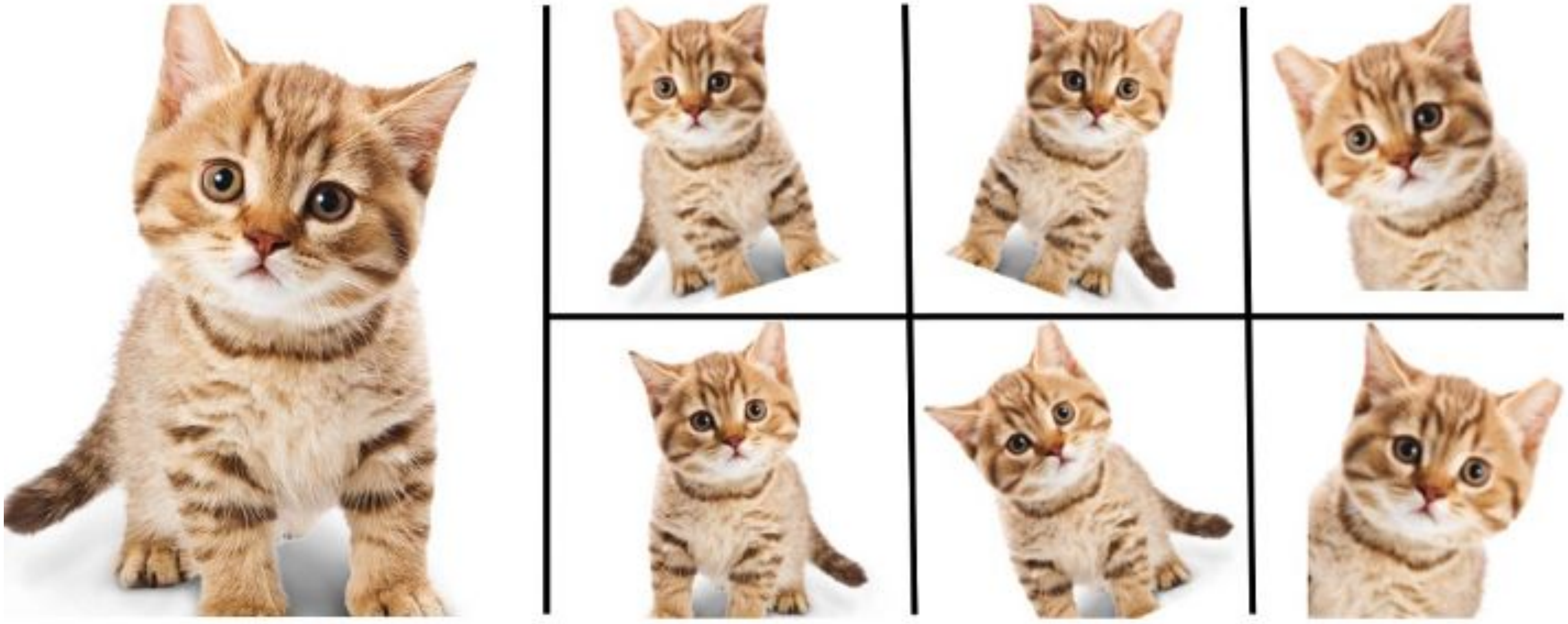
- **Code**
  - finetune_ptl.ipynb
- **Data**
  - Set image dimensions & location
  - Read images from folder in batches
- **Model**
  - Load trained model from feature extraction code
  - Weights in last few convolutional blocks and top model will be adjusted during training
  - All other weights in pre-trained model are frozen
- **Train model**
  - Use training data to adjust top model weights
  - Use validation data to determine when to stop training
- **Evaluate model**
  - Calculate accuracy, etc.
  - Perform inference on test images

Early Stopping

Using validation data to determine when to stop training to avoid overfitting



Data Augmentation



Add variability to your dataset

https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

RESOURCES

- Transfer Learning
  - http://cs231n.github.io/transfer-learning/
- ImageNet
  - http://www.image-net.org/
- PyTorch Lightning
  - https://lightning.ai/docs/pytorch/stable/