SDSC HPC/DS Summer Institute 2024
Deep Learning - Transfer Learning

SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

Deep Learning Agenda

8:00 - 8:05 – Welcome
8:05 - 9:05 – Intro to NN/CNN
9:05 - 9:15 – Break
9:15 - 10:15 – Deep Learning
10:15 - 11:00 – DL Layers & Architectures
11:00 - 11:30 – Break/Lunch
11:30 - 12:30 – DL Transfer Learning
12:30 - 12:40 – Break
12:40 - 1:40 – DL Other Topics
1:40 - 2:00 – Wrapup

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

Deep Learning
Transfer Learning

Mai H. Nguyen, Ph.D.

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

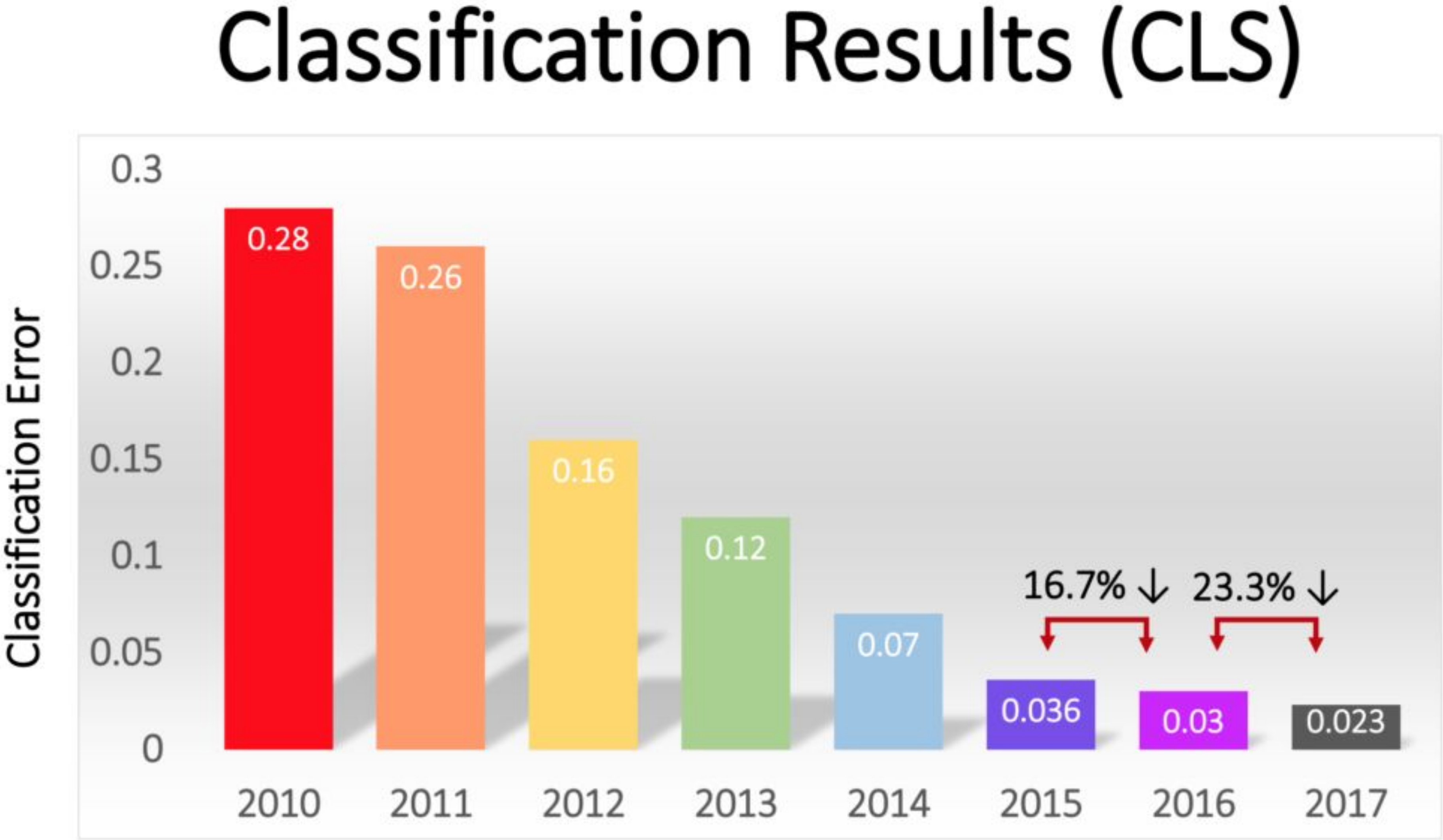
CNNs for Transfer Learning

- **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)

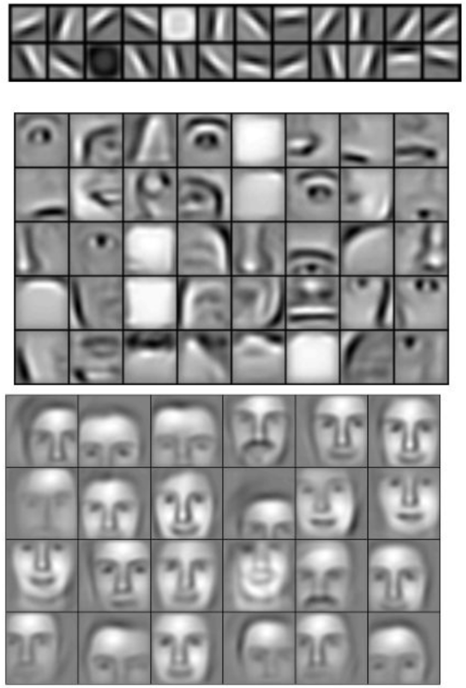
Results on ImageNet Classification



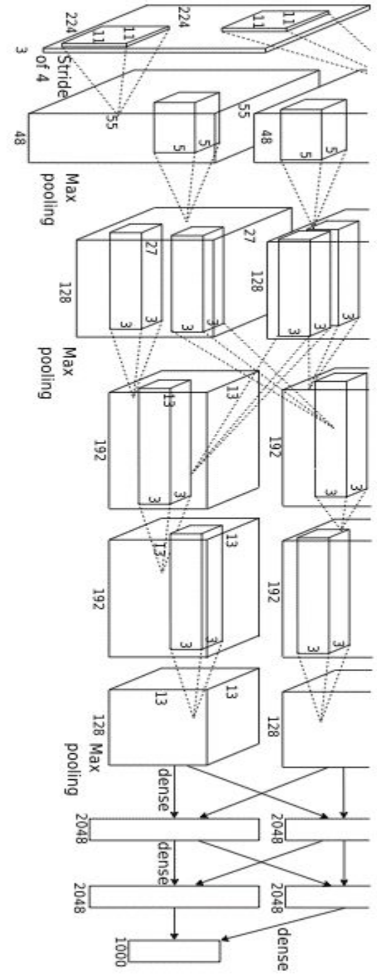
Transfer Learning

Input

Learned hierarchy

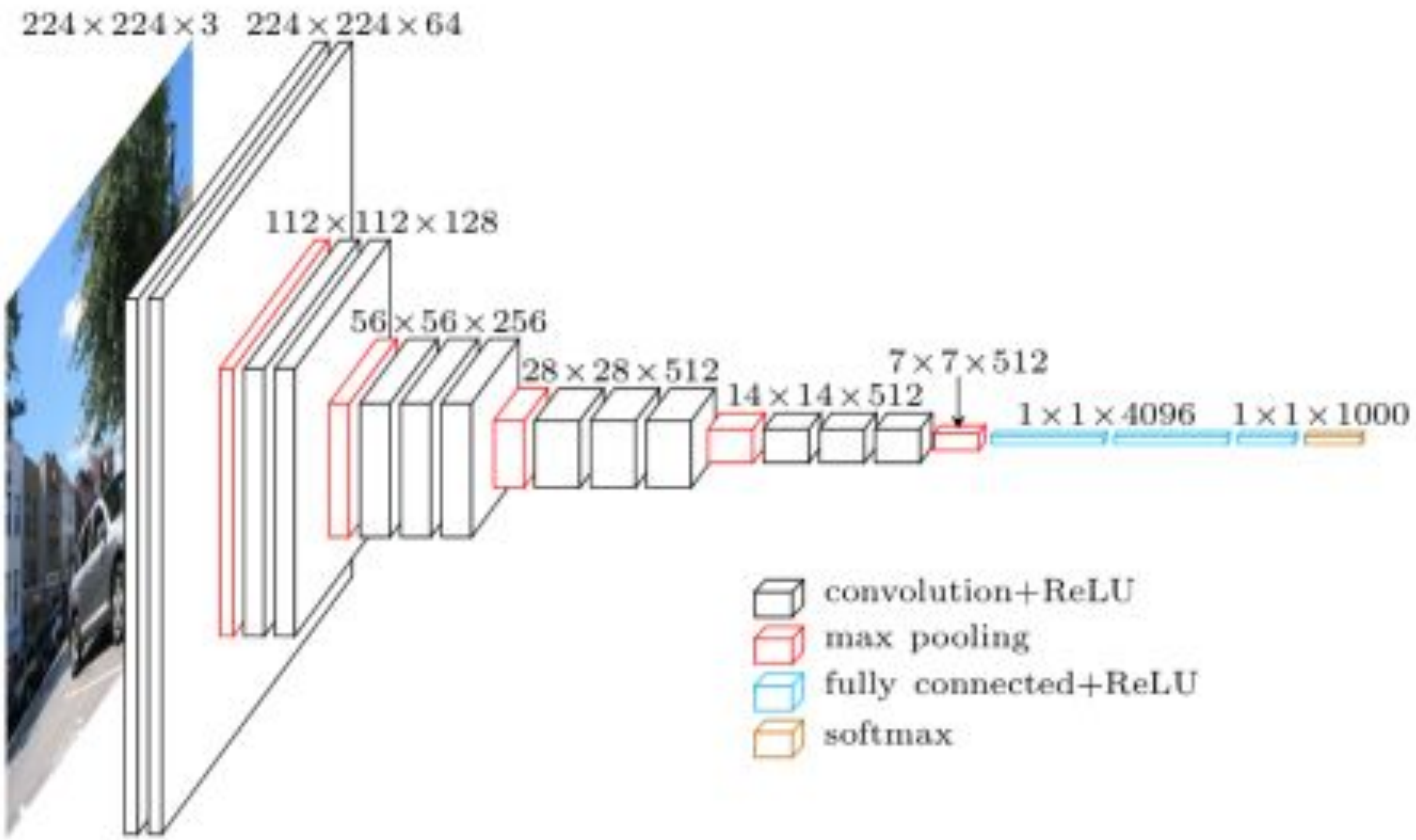


Output



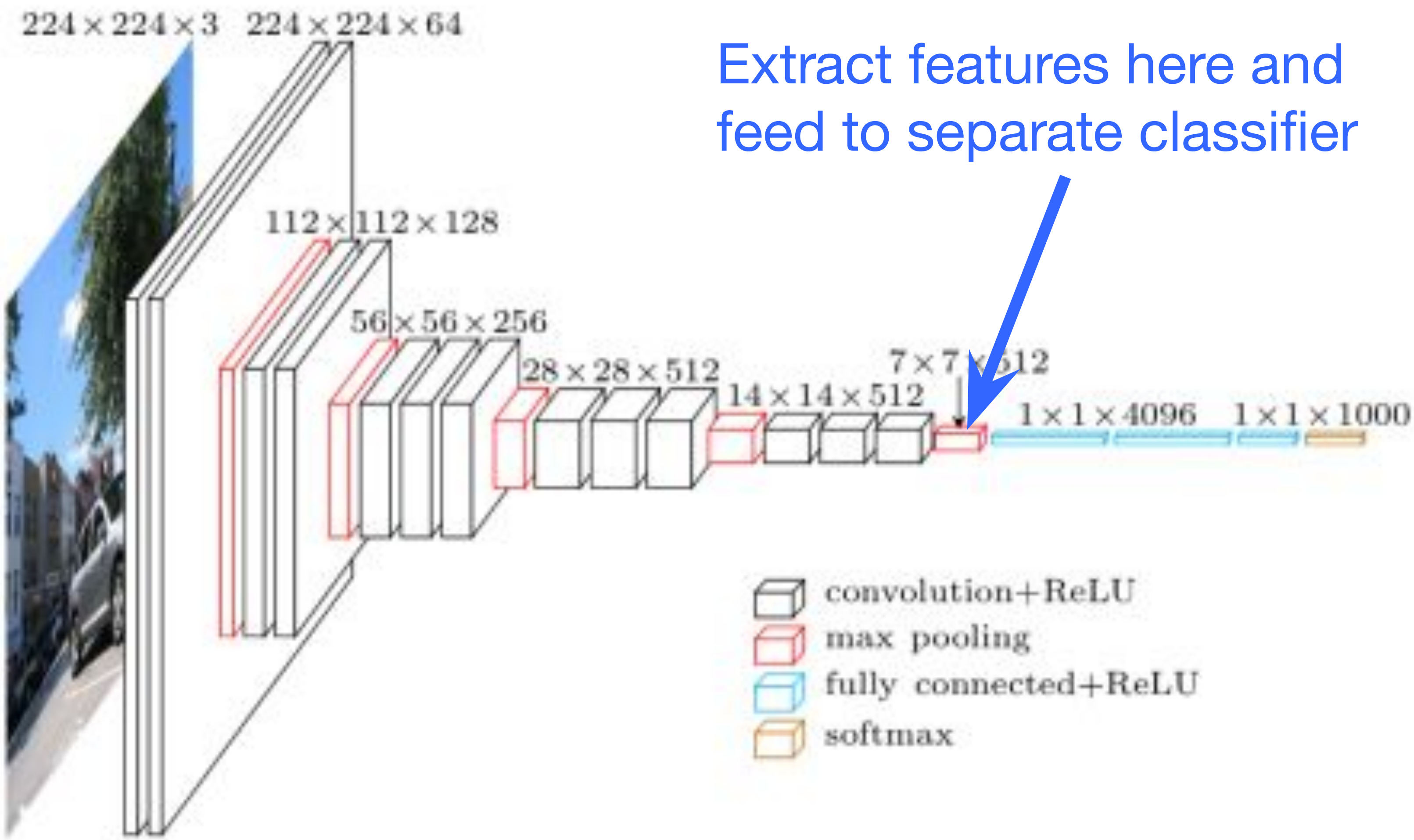
Lee et al. 'Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations' ICML 2009

Pre-Trained Model



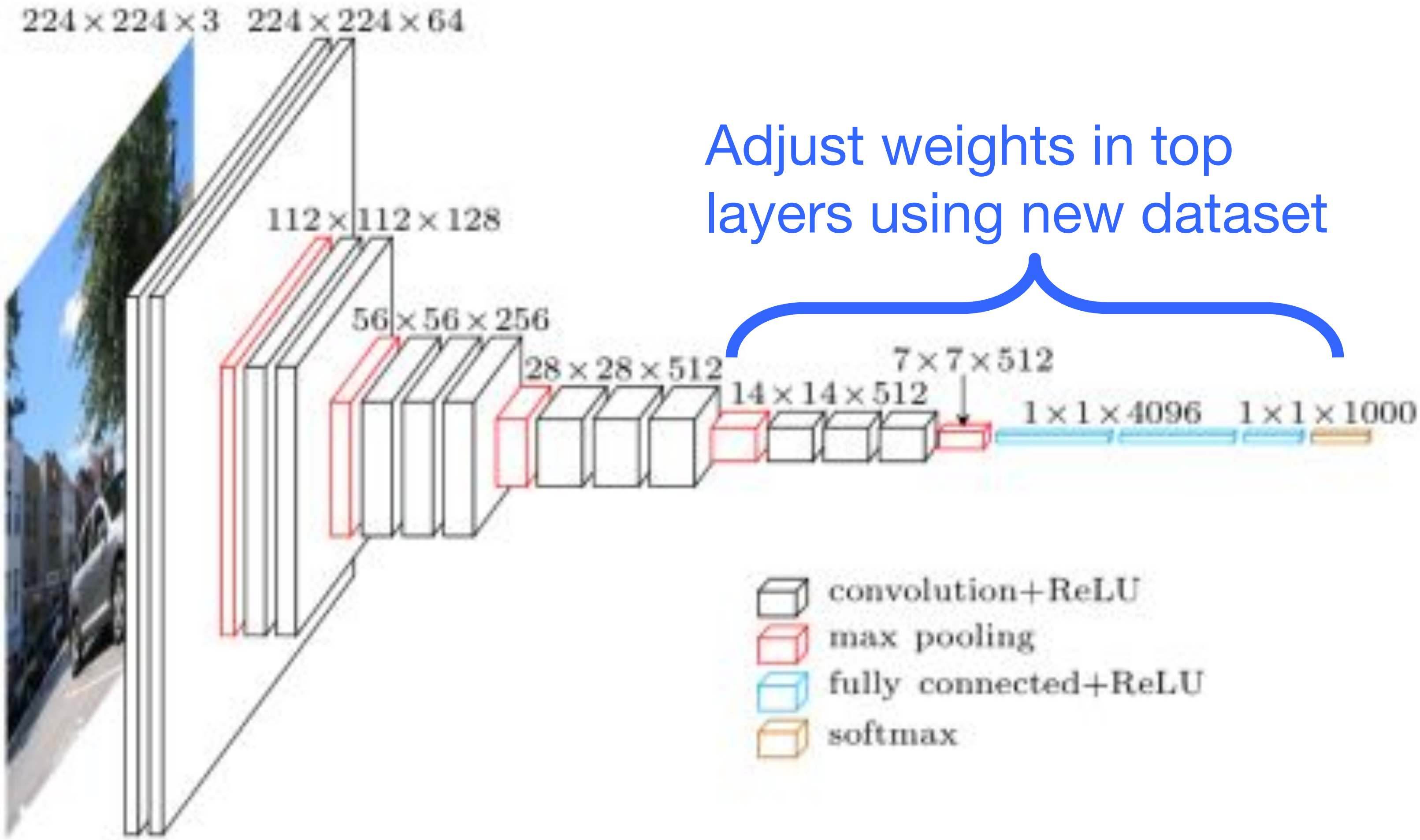
https://www.cs.toronto.edu/~frossard/post/vgg16/

Transfer Learning - Feature Extraction



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/

When & How to Fine Tune

- **New dataset is small & similar to original dataset**
  - Extract features from higher layer and feed to separate classifier
- **New dataset is large & similar to original dataset**
  - Fine tune top or all layers
- **New dataset is small & different from original dataset**
  - Extract features from lower layer and feed to separate classifier
- **New dataset is large & different from original dataset**
  - Fine tune top or all layers

http://cs231n.github.io/transfer-learning/

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
    - Code: feature_extract.ipynb, feature_extract_soln.ipynb
  - Fine tune
    - Adjust weights of last few layers of pre-trained CNN and top classifier model through training
    - Code: finetune.ipynb, finetune_soln.ipynb
  - Note
    - Shut down kernel for feature_extract.ipynb before running finetune.ipynb to avoid out-of-memory errors (Kernel -> Shut Down Kernel)

Data

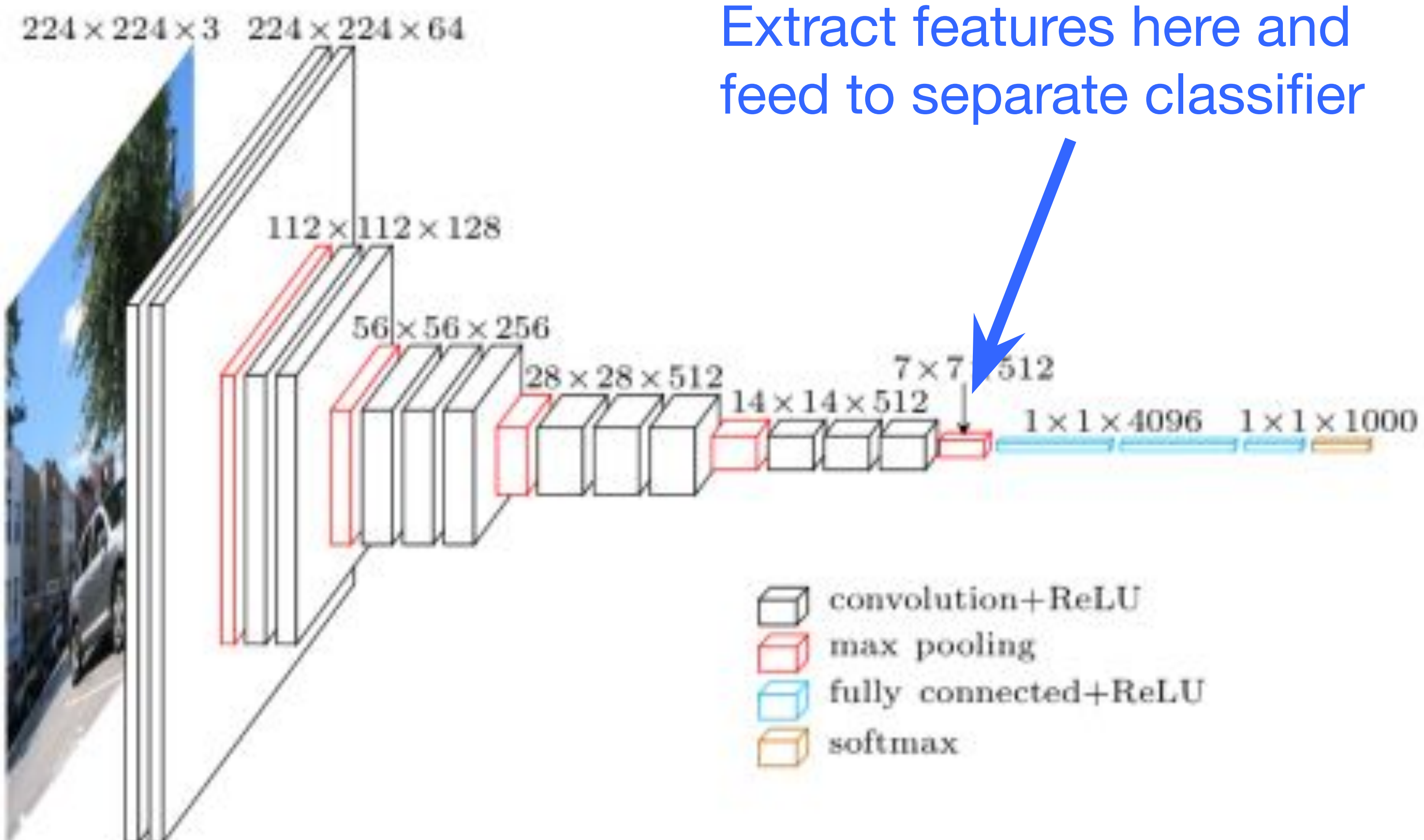
- **Subset of Dogs Vs. Cats dataset from Kaggle**
  - https://www.kaggle.com/c/dogs-vs-cats
- **Train**
  - 1000 cats + 1000 dogs
- **Validation**
  - 200 cats + 200 dogs
- **Test**
  - 200 cats + 200 dogs

TRANSFER LEARNING - FEATURE EXTRACTION

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

- **Method**
  - Use VGG16 trained on ImageNet data as pre-trained model. Remove last fully connected layer.
  - Extract features from pre-trained model and save
  - Neural network then trained on extracted features to classify cats vs. dogs

TRANSFER LEARNING - FEATURE EXTRACTION



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

Feature Extraction Overview

- **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

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-2024.git

Server Setup for TensorFlow - Portal

- **Expanse Portal**
  - https://portal.expanse.sdsc.edu
- **Parameters**
  - Account: sds184
  - Time limit (min): 180
  - Number of cores: 10
  - Memory required per node: 93 GB
  - GPUs: 1
  - Singularity image: /cm/shared/apps/containers/singularity/ciml/2021/tensorflow-latest.sif
  - Environment module: singularitypro
  - Reservation: ciml-day3
  - Working directory: home
  - Type: JupyterLab

Server Setup for TensorFlow - Command Line

- **In terminal window**
  - jupyter-gpu-shared-tensorflow
    - Alias for:
    - galyleo launch --account ${SI24_ACCOUNT} --reservation ${SI24_RES_GPU} --partition gpu-shared --qos ${SI24_QOS_GPU} --cpus 10 --memory 92 --gpus 1 --time-limit 04:00:00 --env-modules singularitypro --sif ${SI24_CONTAINER_DIR}/tensorflow/tensorflow-latest.sif --bind /cm,/expanse,/scratch --nv --quiet
- **To check queue**
  - squeue -u $USER

Data Setup

- **In terminal window in Jupyter Lab, do the following**
- **Go to your home directory**

  cd

  pwd   # Should see /home/$USER

- **Get data**

  Don't forget the period at the end!

  - mkdir data   # If doesn't already exist
  - cd data
  - cp /cm/shared/examples/sdsc/ciml/2022/catsVsDogs.zip .
  - unzip -q catsVsDogs.zip
  - ls catsVsDogs   # Should see train, val, test

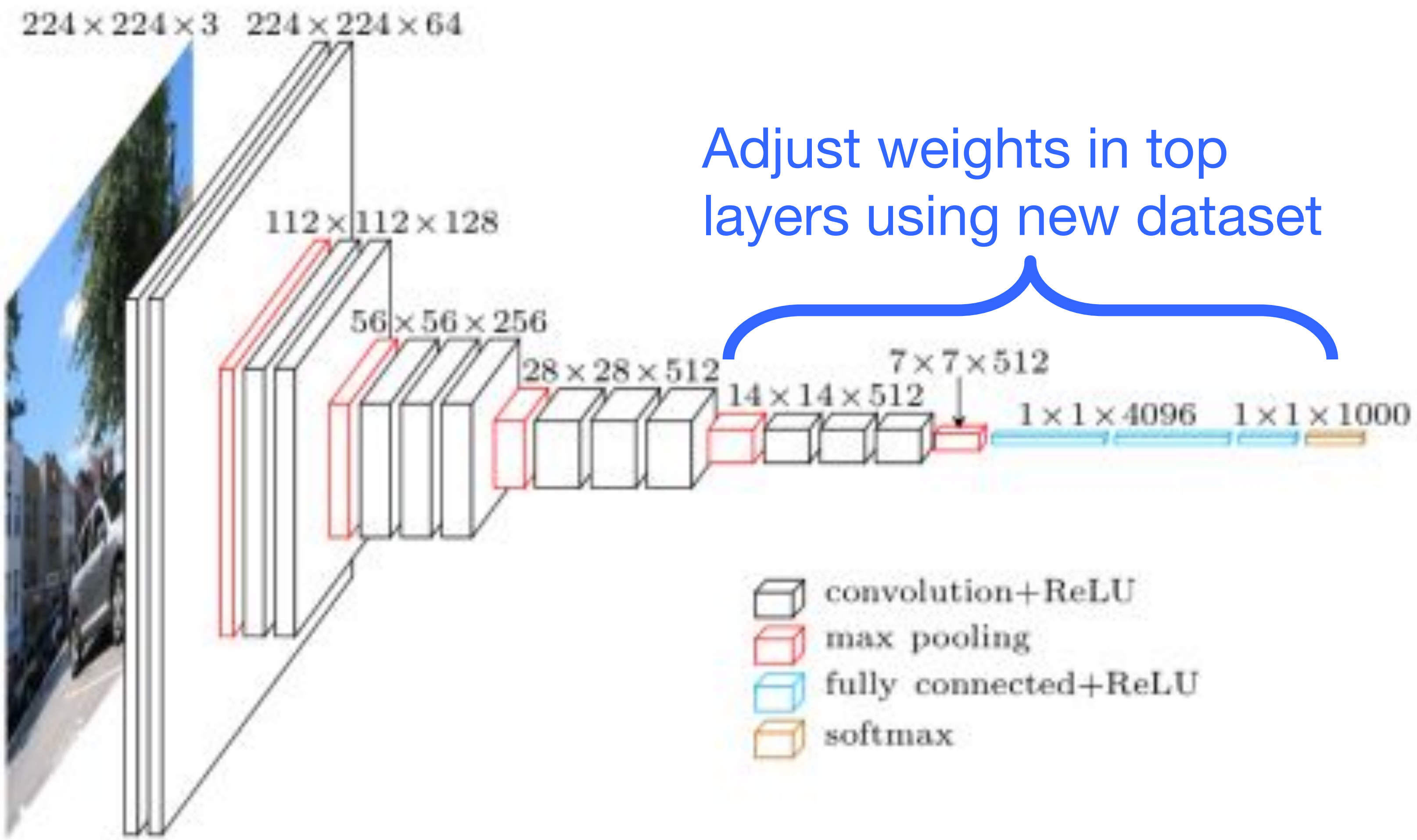
Data

- **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 - FINE TUNING

- **Data**
  - Cats and dogs images from Kaggle
- **Method**
  - Use VGG16 trained on ImageNet data as pre-trained model.
  - Replace last fully connected layer with neural network trained from Feature Extraction hands-on.
  - Fine tune last convolution block and fully connected layer.

TRANSFER LEARNING - FINE TUNING



Adjust weights in top layers using new dataset

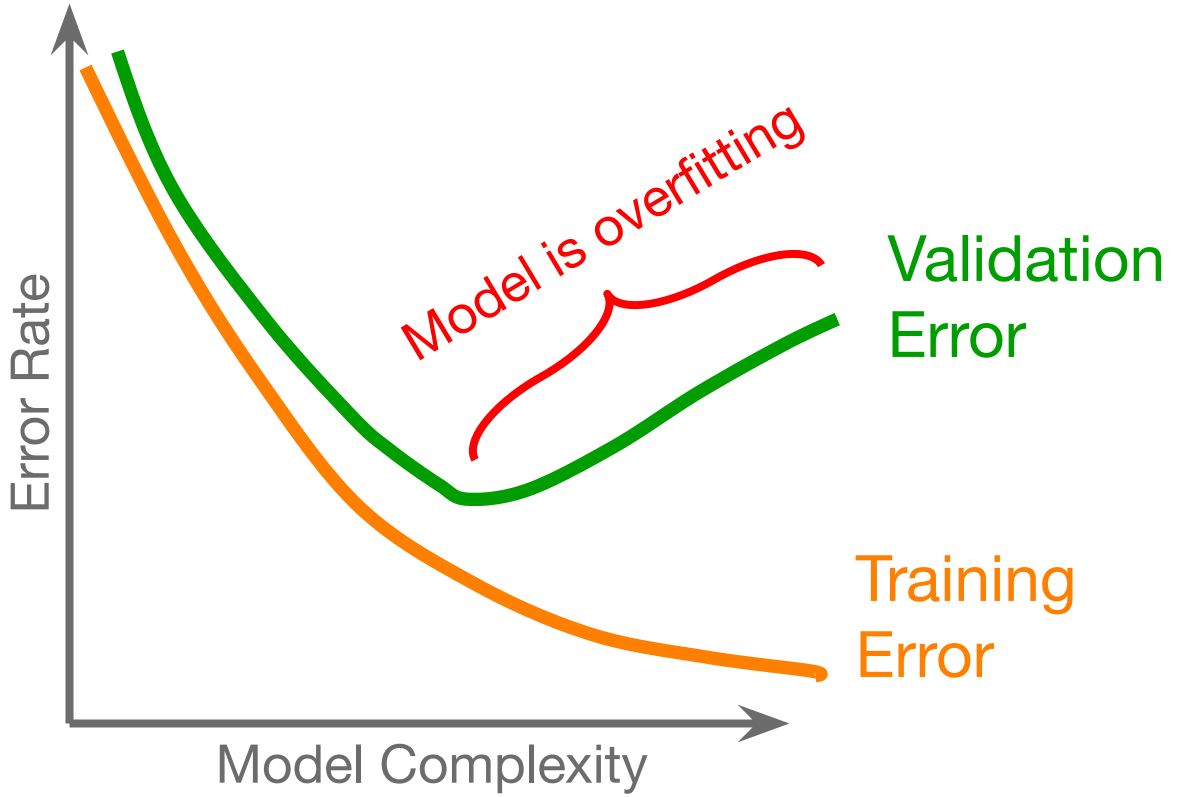
Source: https://www.cs.toronto.edu/~frossard/post/vgg16/

Fine Tune Overview

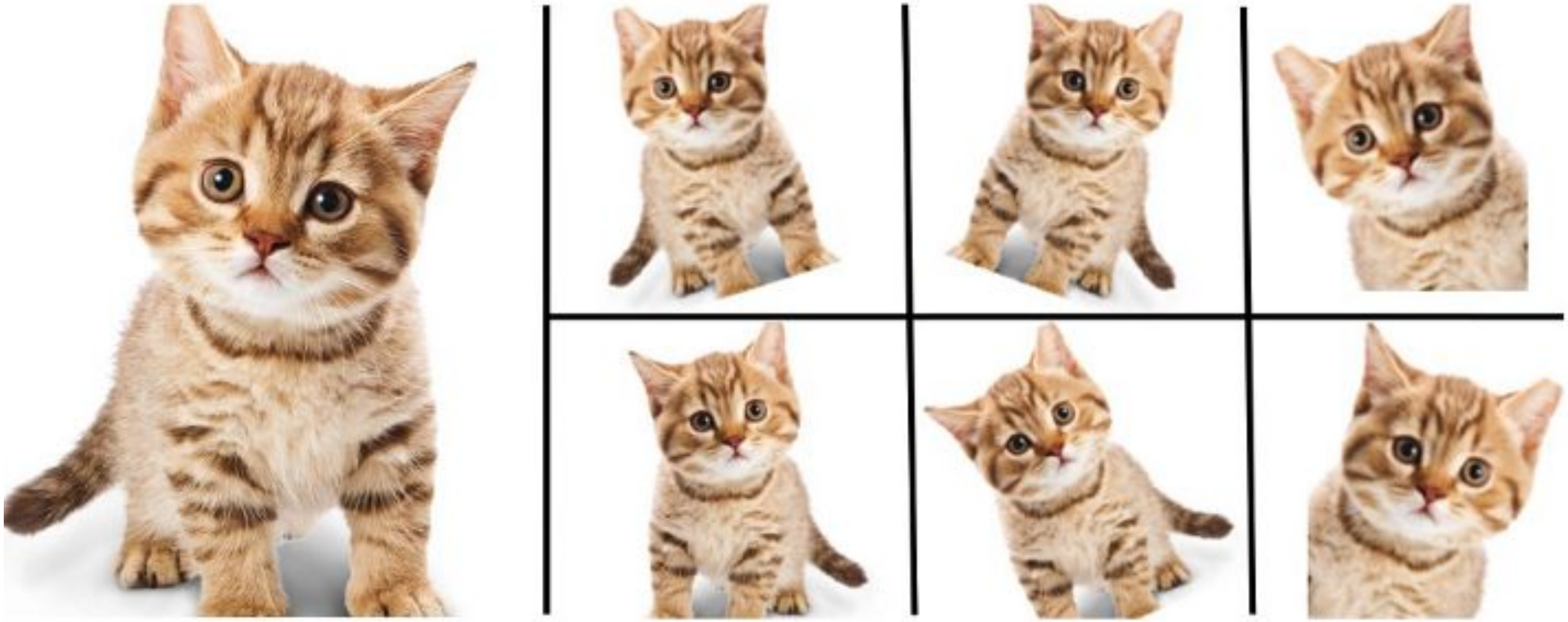
- **Data**
  - Set image dimensions & location
  - Read images from folder in batches
- **Model**
  - Load trained model from feature extraction code
  - Weights in last few convolutional blocks and top model will be adjusted during training
  - All other weights in pre-trained model are frozen
- **Train 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



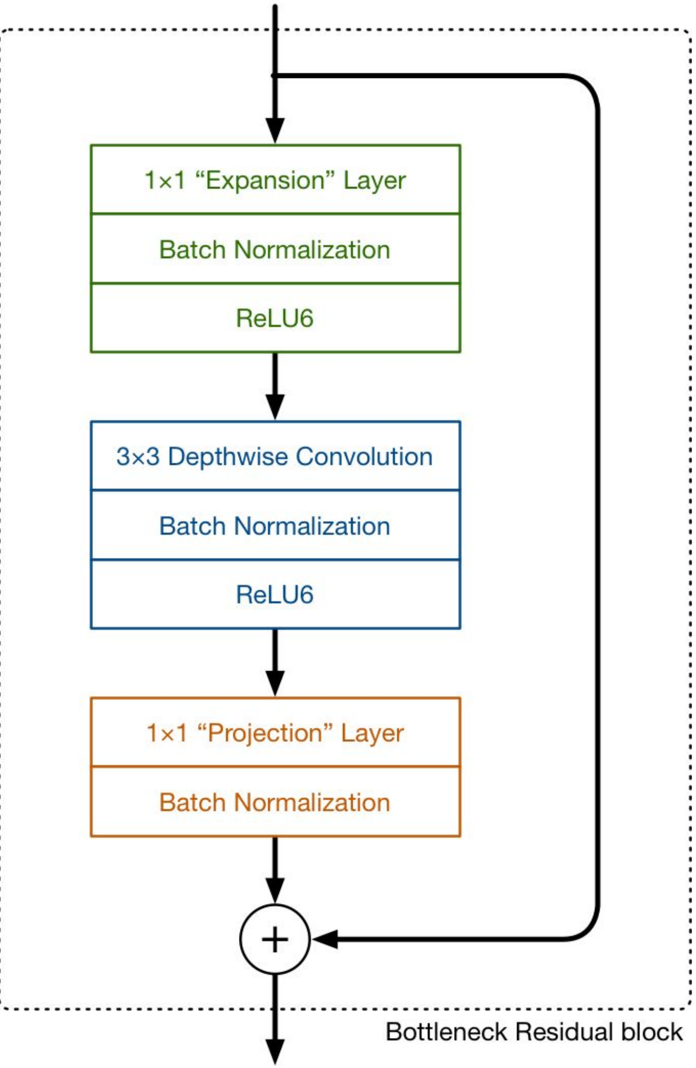
Data Augmentation



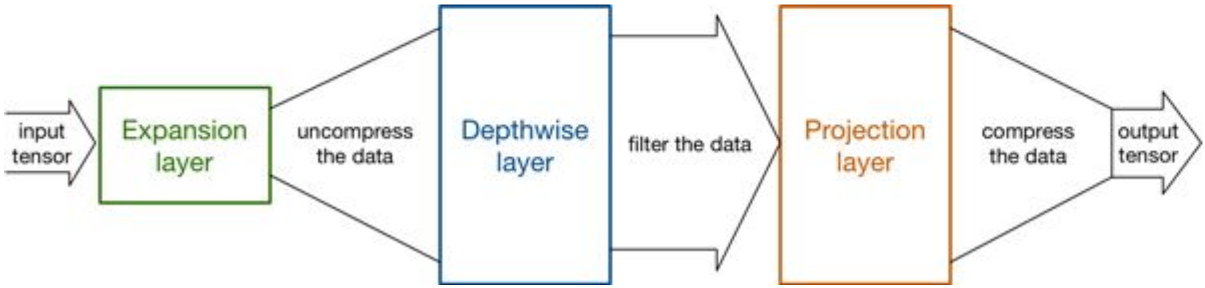
Add variability to your dataset

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

MobileNetV2



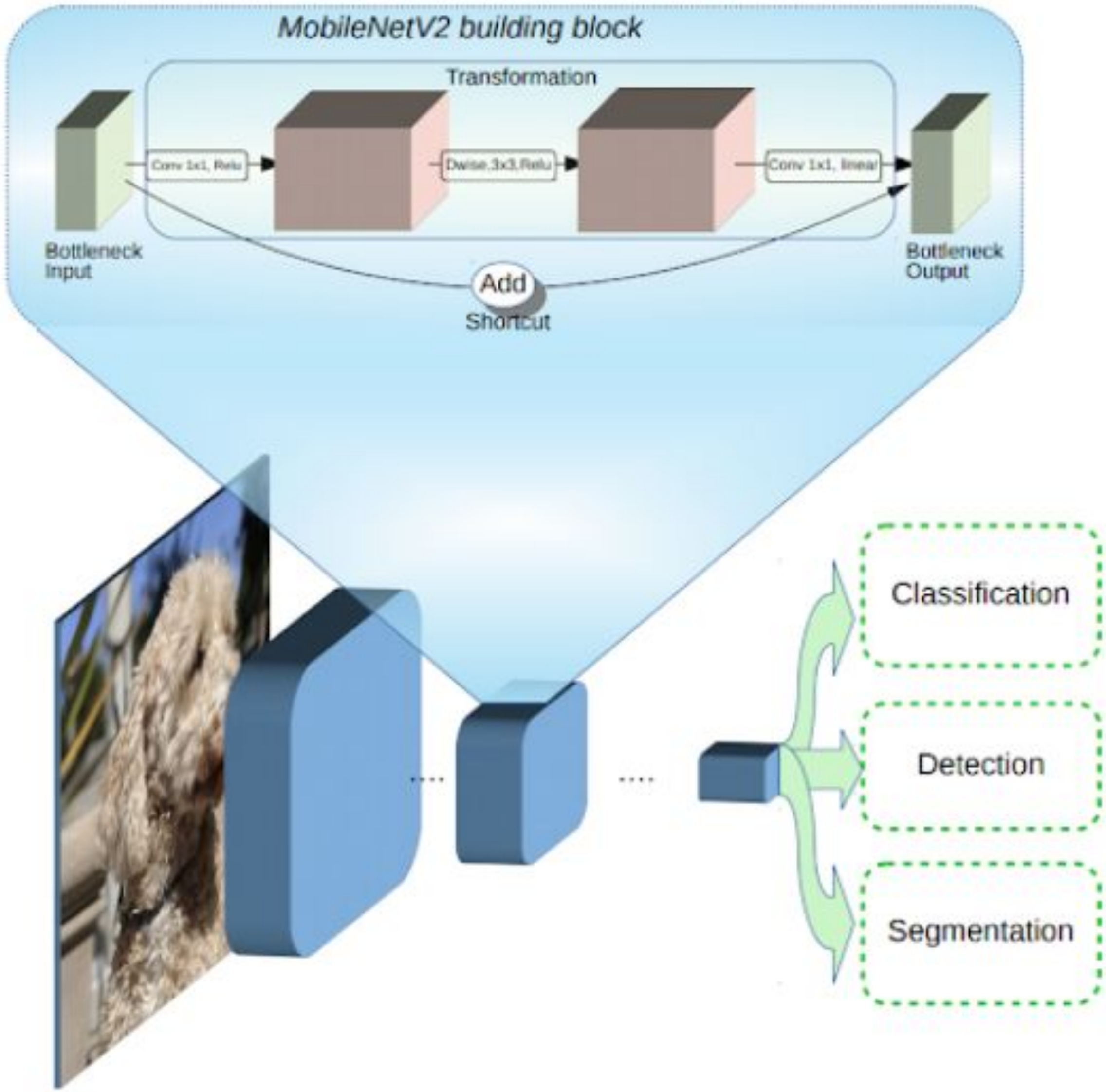
- CNN
- Lightweight architecture
- Designed for mobile devices



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

MobileNetV2

- CNN
- Lightweight architecture
- Designed for mobile devices



RESOURCES

- **TensorFlow Tutorial on Transfer Learning**
  - https://www.tensorflow.org/tutorials/images/transfer_learning
- **Transfer Learning**
  - http://cs231n.github.io/transfer-learning/
- **ImageNet**
  - http://www.image-net.org
- **TensorFlow/Keras API**
  - https://www.tensorflow.org/api_docs/python/tf/keras/Model