

# HPC & Data Science Summer Institute 2023

## Deep Learning



# 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**

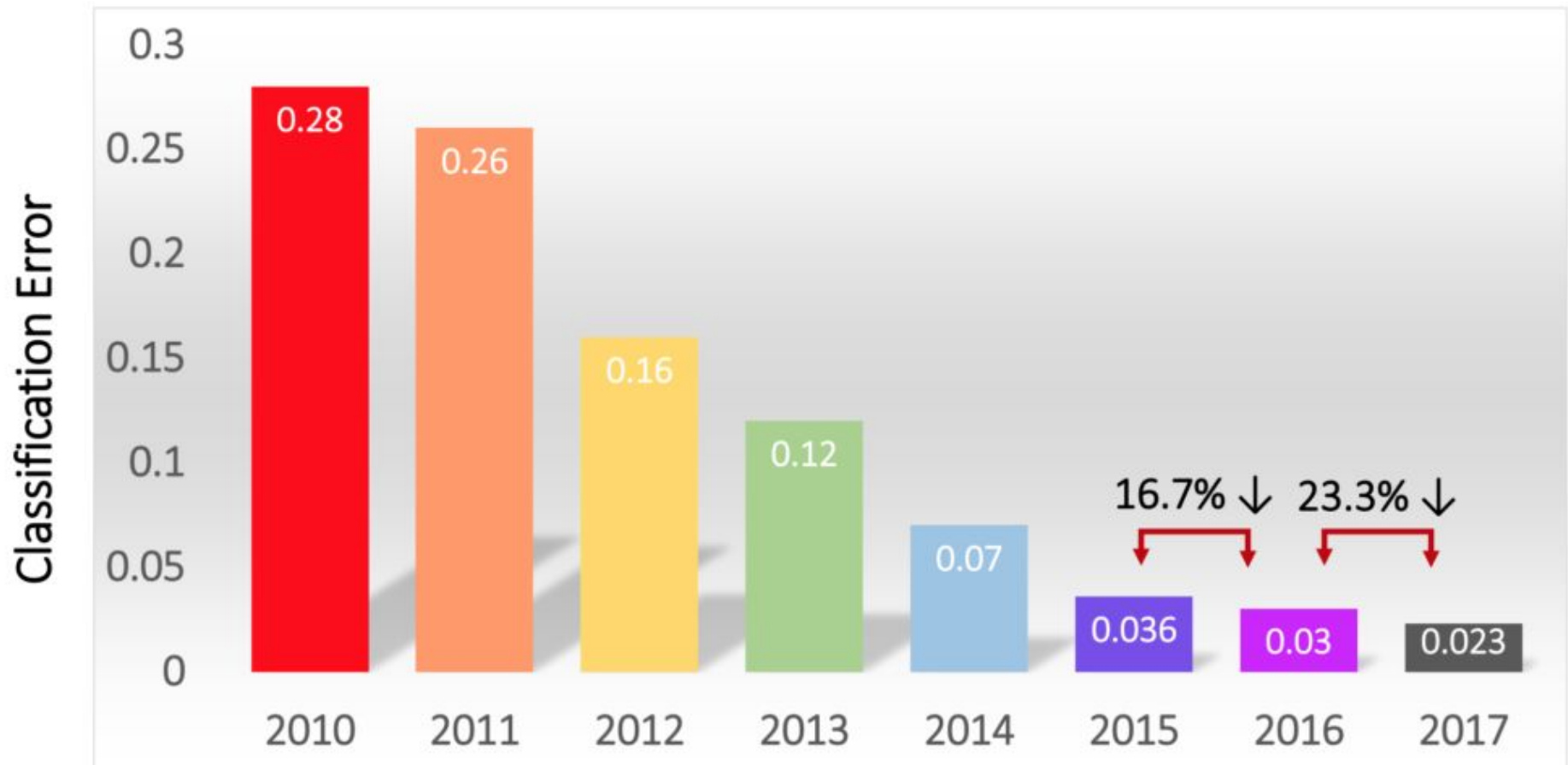
- Developed for computer vision research
- ~14,000,000 images hand-annotated
- ~22,000 categories

- **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

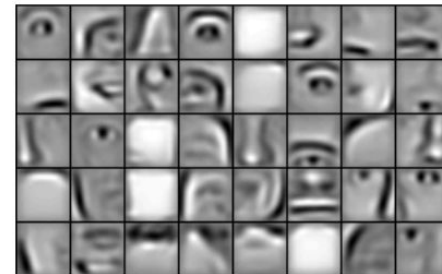
## Classification Results (CLS)





# Transfer Learning

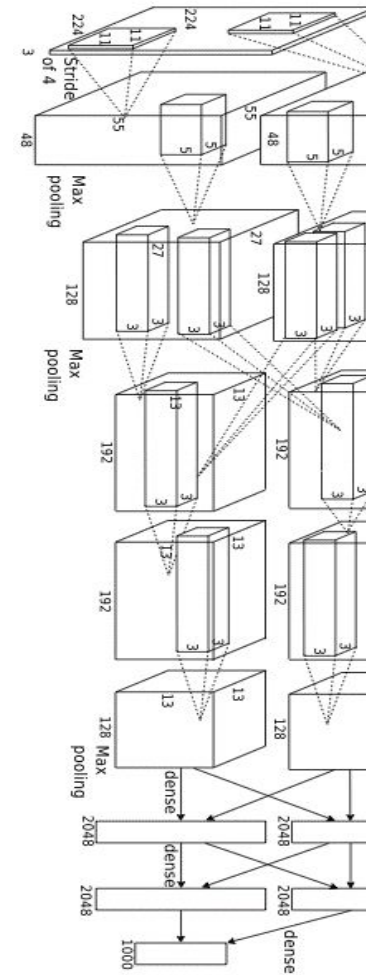
*Input*



*Learned  
hierarchy*

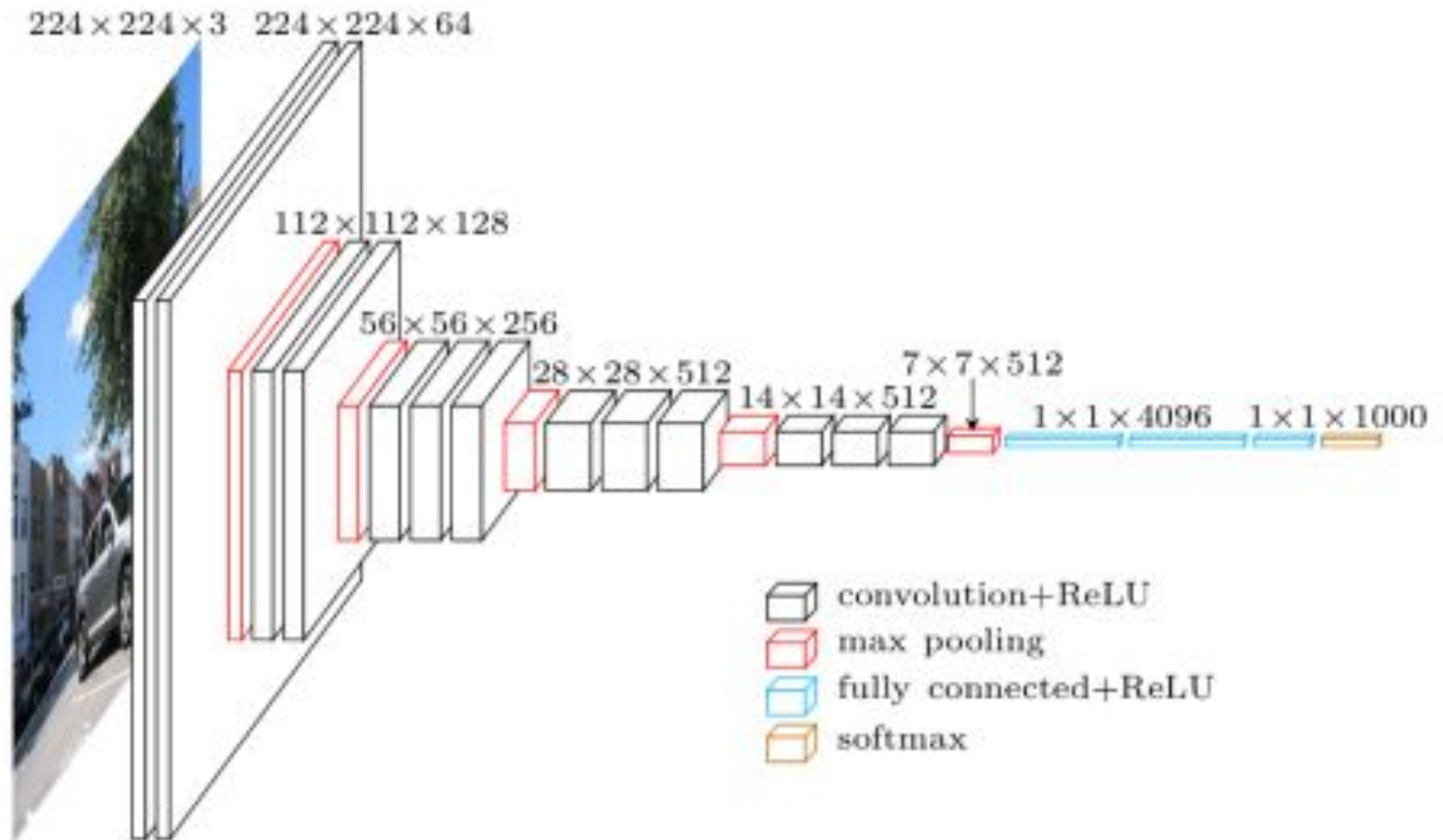


*Output*



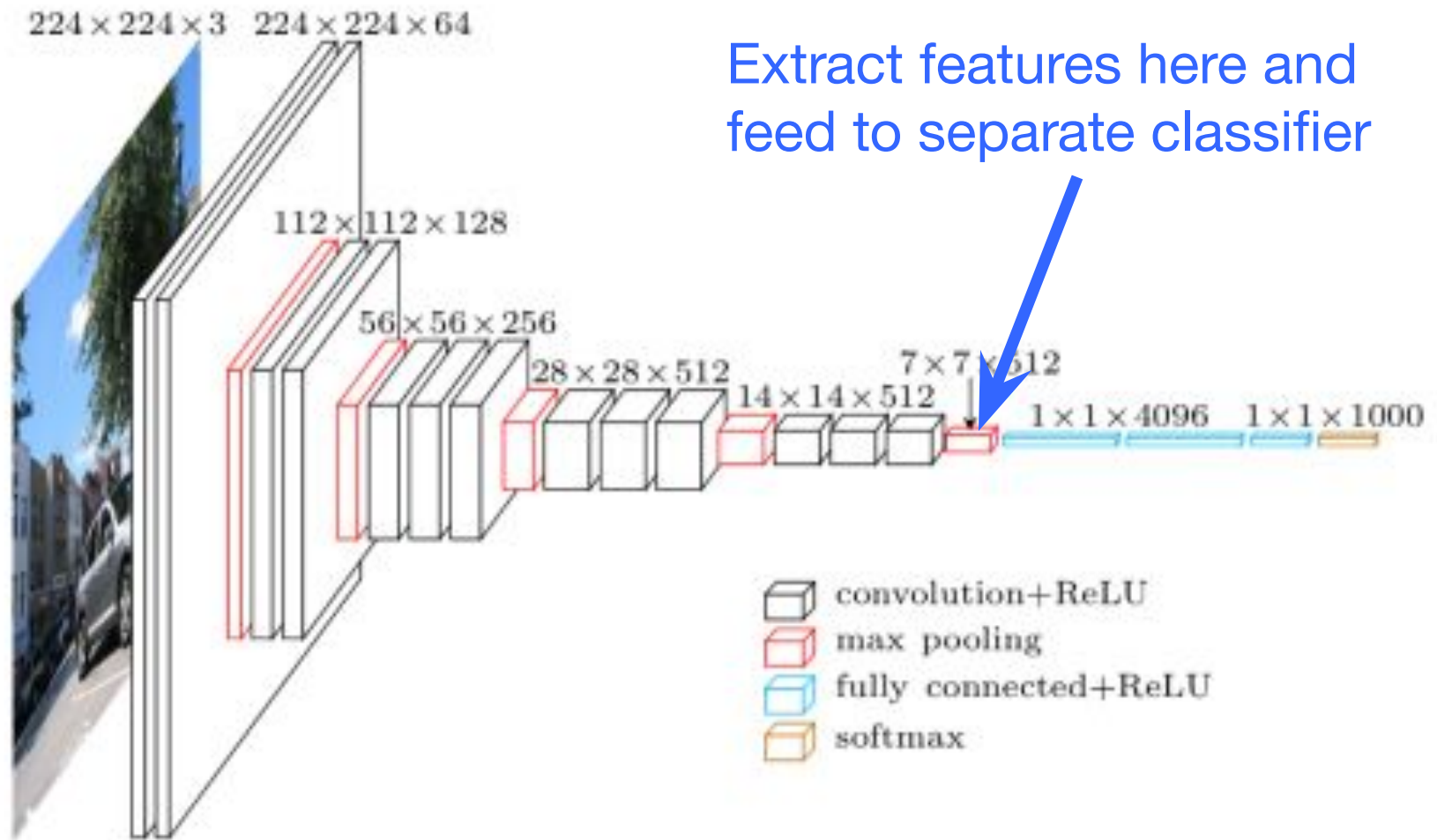
*Lee et al. 'Convolutional Deep Belief Networks for Scalable  
Unsupervised Learning of Hierarchical Representations' ICML 2009*

# Pre-Trained Model



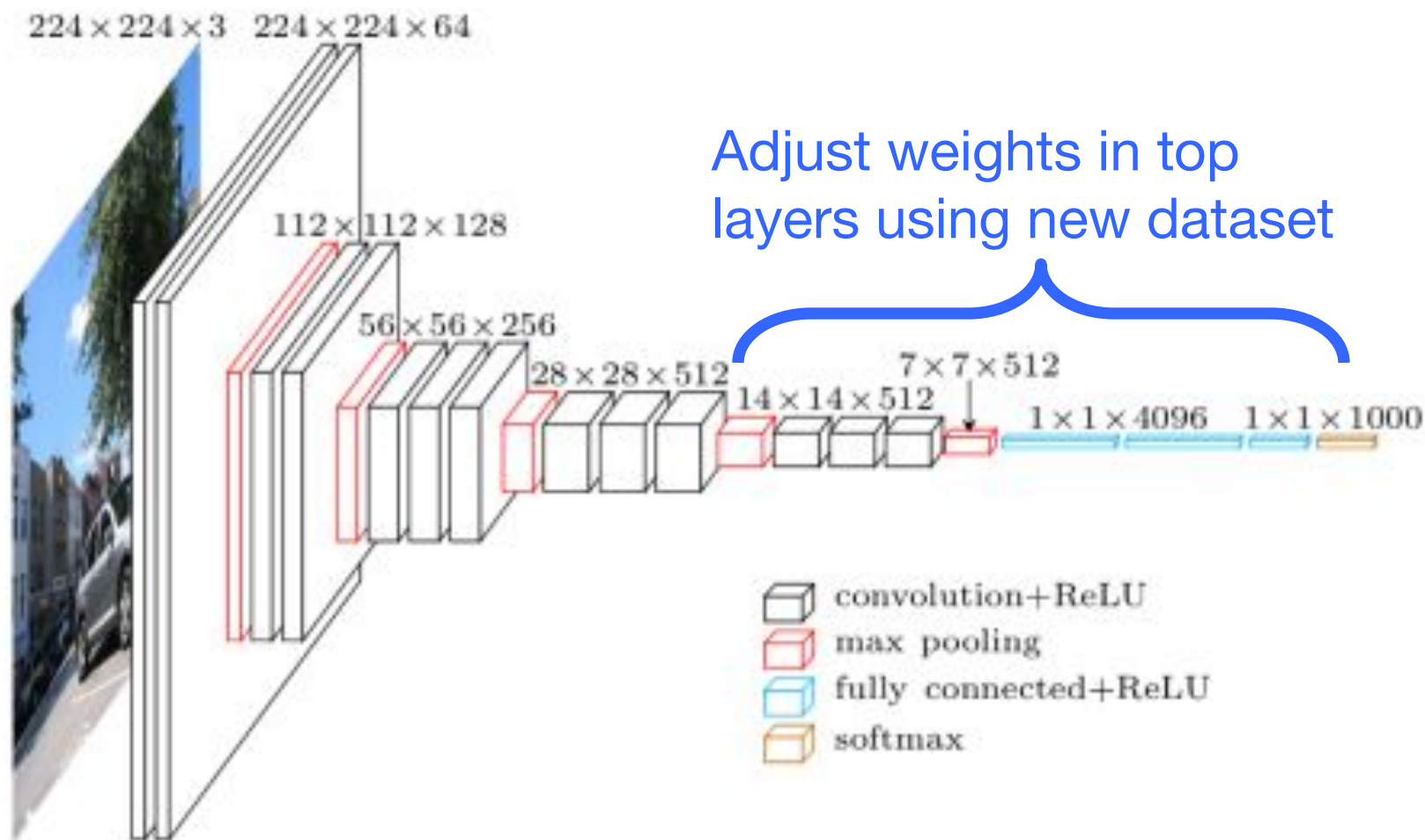
<https://www.cs.toronto.edu/~frossard/post/vgg16/>

# Transfer Learning - Feature Extraction



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

# Transfer Learning - Fine Tuning



<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 extract features
- Fine tune
  - Adjust weights of last few layers of pre-trained CNN 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-2023>



# Server Setup for TensorFlow - Command Line

- **In terminal window**
  - `jupyter-gpu-shared-tensorflow`
    - Alias for:
    - `galileo launch --account ${CIML23_ACCOUNT} --reservation ${CIML23_RES_GPU} --partition gpu-shared --qos ${CIML23_QOS_GPU} --cpus 10 --memory 92 --gpus 1 --time-limit 04:00:00 --env-modules singularitypro --sif ${CIML23_CONTAINER_DIR}/tensorflow/tensorflow-latest.sif --bind /cm,/expance,/scratch --nv --quiet`
- **To check queue**
  - `squeue -u $USER`

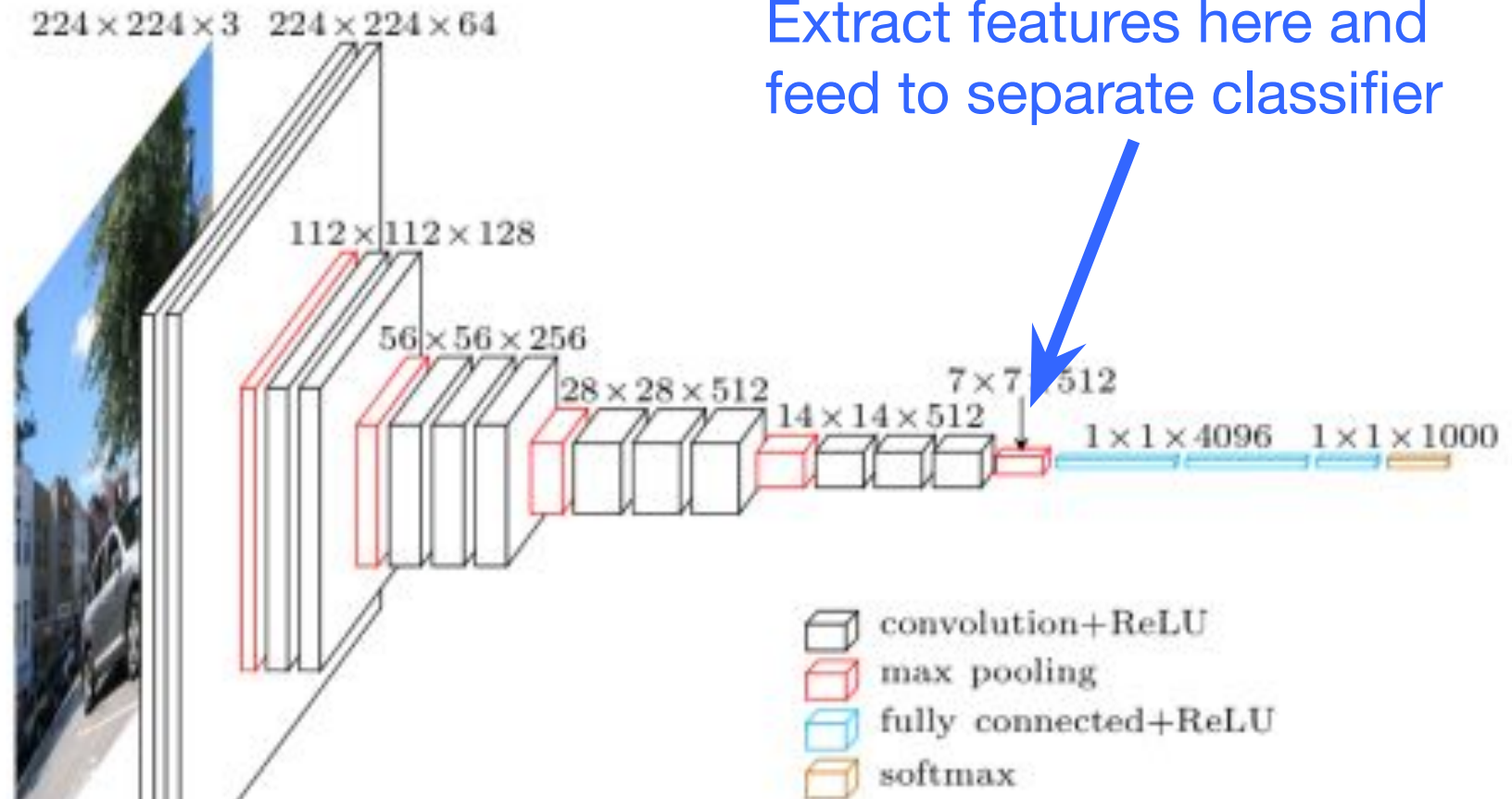
# Data

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

# Code

- **features\_extract.ipynb**
  - Transfer learning with feature extraction
- **finetune.ipynb**
  - Transfer learning with fine tuning
- **Note**
  - Restart kernel for features\_extract\_tf.ipynb before running finetune\_tf.ipynb to avoid out-of-memory errors

# TRANSFER LEARNING - FEATURE EXTRACTION

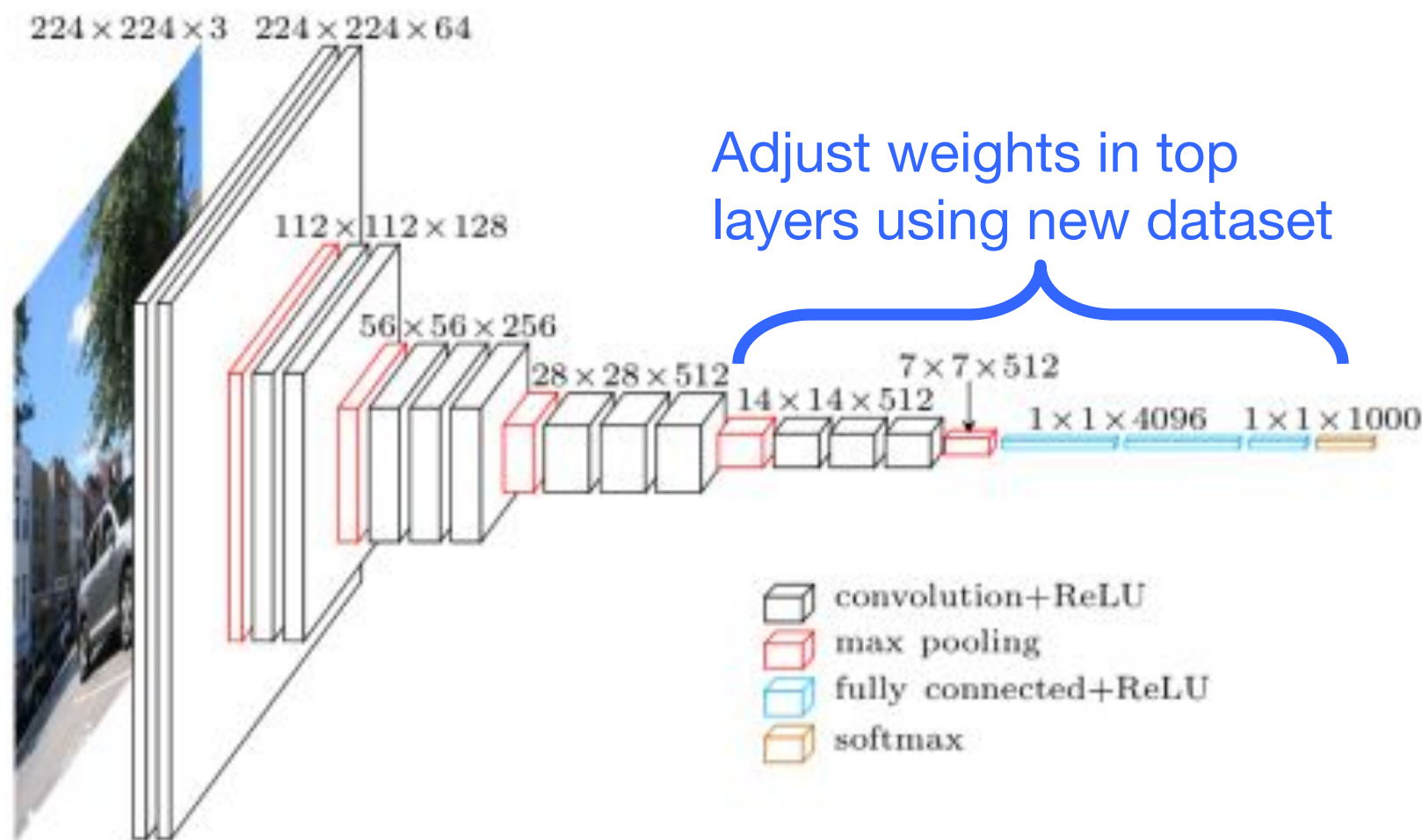


Source: <https://www.cs.toronto.edu/~frossard/post/vgg16/>

# Feature Extraction Overview

- **Data**
  - Set image dimensions & location
  - Use ImageDataGenerator to read images from folder
- **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



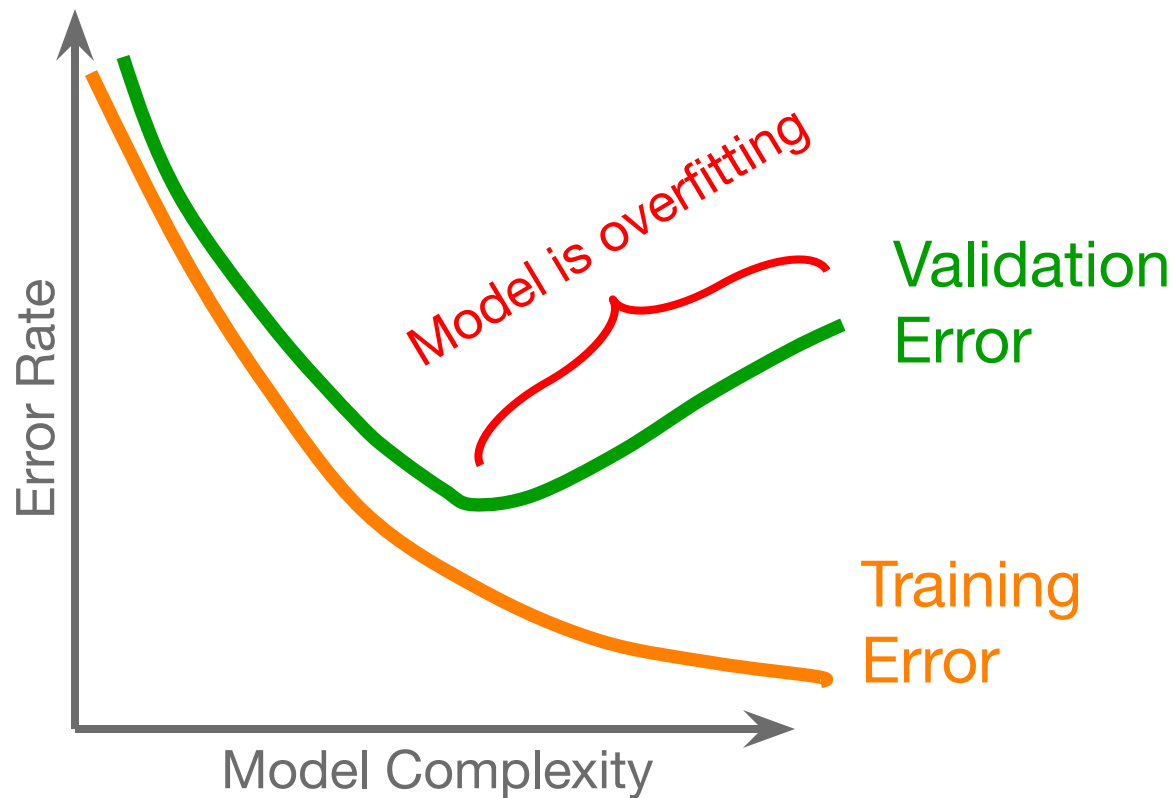
Source: <https://www.cs.toronto.edu/~frossard/post/vgg16/>

# Fine Tune Overview

- **Data**
  - Set image dimensions & location
  - Use ImageDataGenerator to read images from folder
- **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

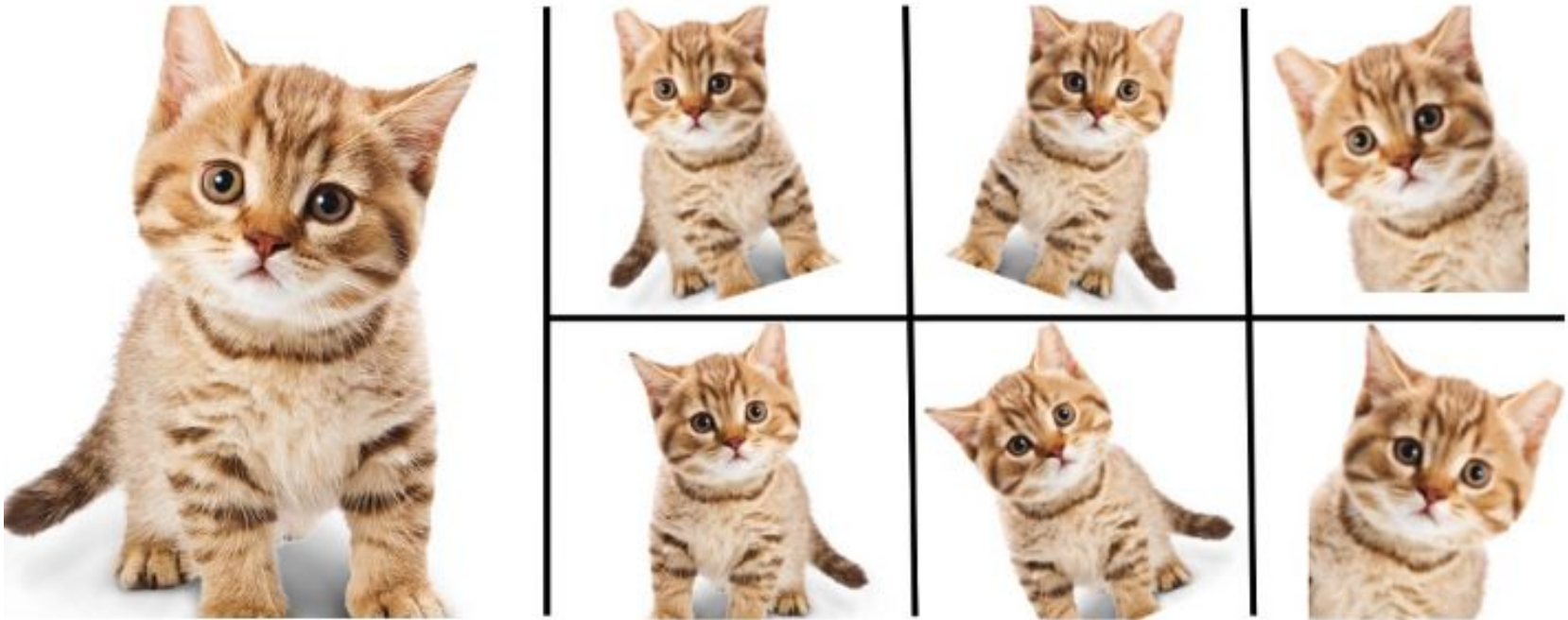
# Early Stopping

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





# Data Augmentation

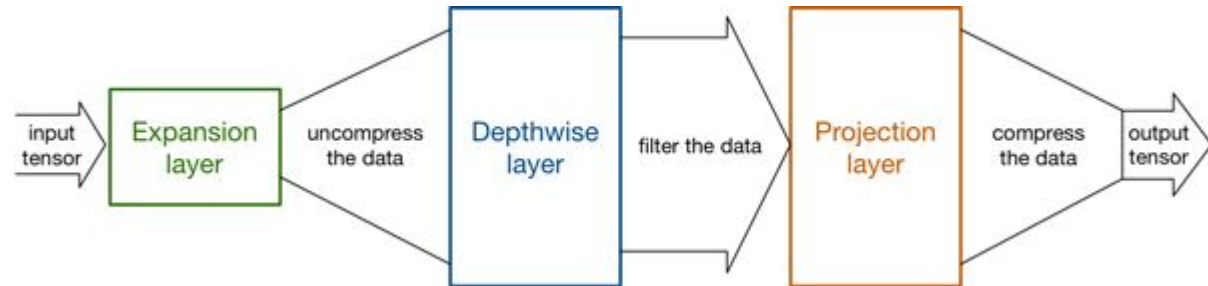
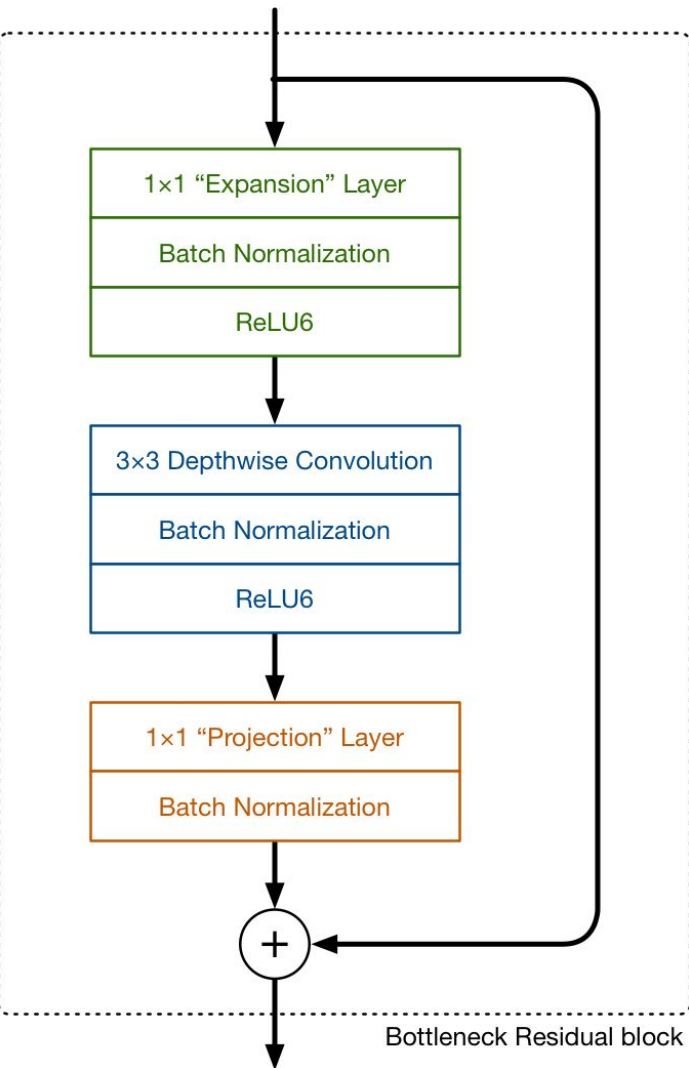


Add variability to your dataset

<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/>

# RESOURCES

- **TensorFlow Tutorial on Transfer Learning**
  - [https://www.tensorflow.org/tutorials/images/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](https://www.tensorflow.org/api_docs/python/tf/keras/Model)