CIML Summer Institute
Hands-On Exercises

SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

Hands-On Exercises

- 8:00 - 8:45 -- Writing & Sharing Computational Analyses in Jupyter Notebooks
- 8:45 - 9:45 -- Spark
- 9:45 - 10:00 -- Break
- 10:00 - 10:30 -- Spark
- 10:30 - 11:00 -- MNIST & TensorBoard
- 11:00 - 12:00 -- Lunch
- 12:00 - 1:00 -- Deep Learning Transfer Learning
- 1:00 - 1:45 -- Deep Sequence Learning
- 1:45 - 2:00 -- Q&A

Deep Learning Transfer Learning Hands-On

Mai H. Nguyen, Ph.D.

WHAT IS 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 last fully connected layer from pre-trained model
  - Treat rest of network as feature extractor
  - Use features to train new classifier ("top model")
- **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

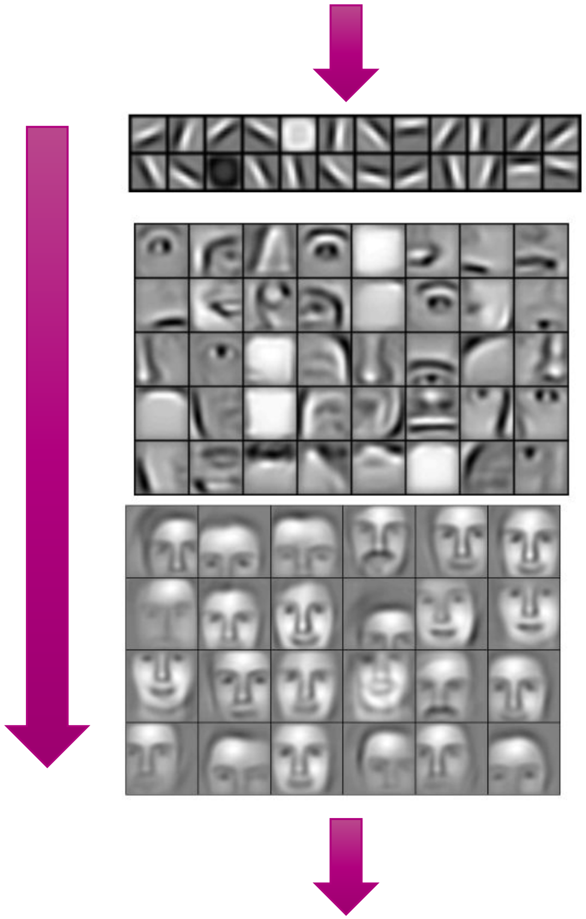
ImageNet

- **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
    - 2011: ~25% (conventional image processing techniques)
    - 2012: 15.3% (AlexNet)
    - 2015: 3.57% (ResNet; better than human performance)
    - 2016: 2.99% (16.7% error reduction)
    - 2017: 2.25% (23.3% error reduction)

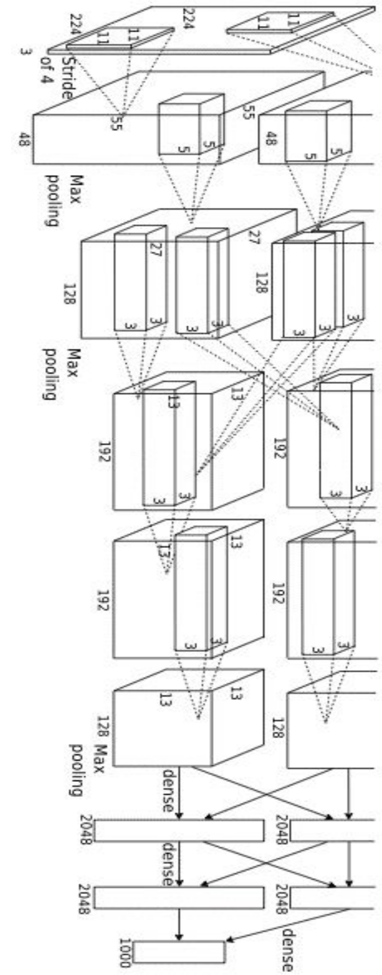
TRANSFER LEARNING

Input

Learned hierarchy

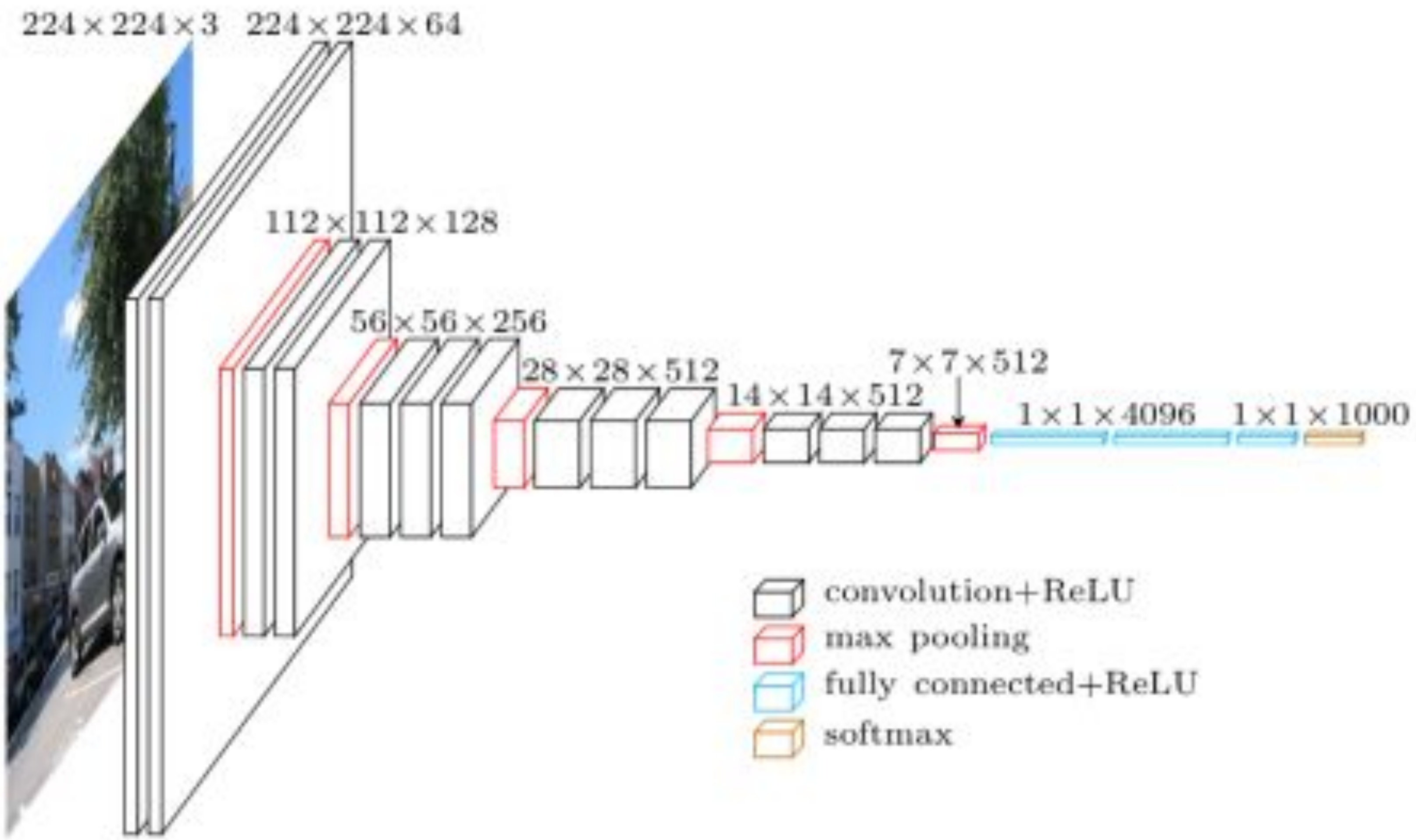


Output



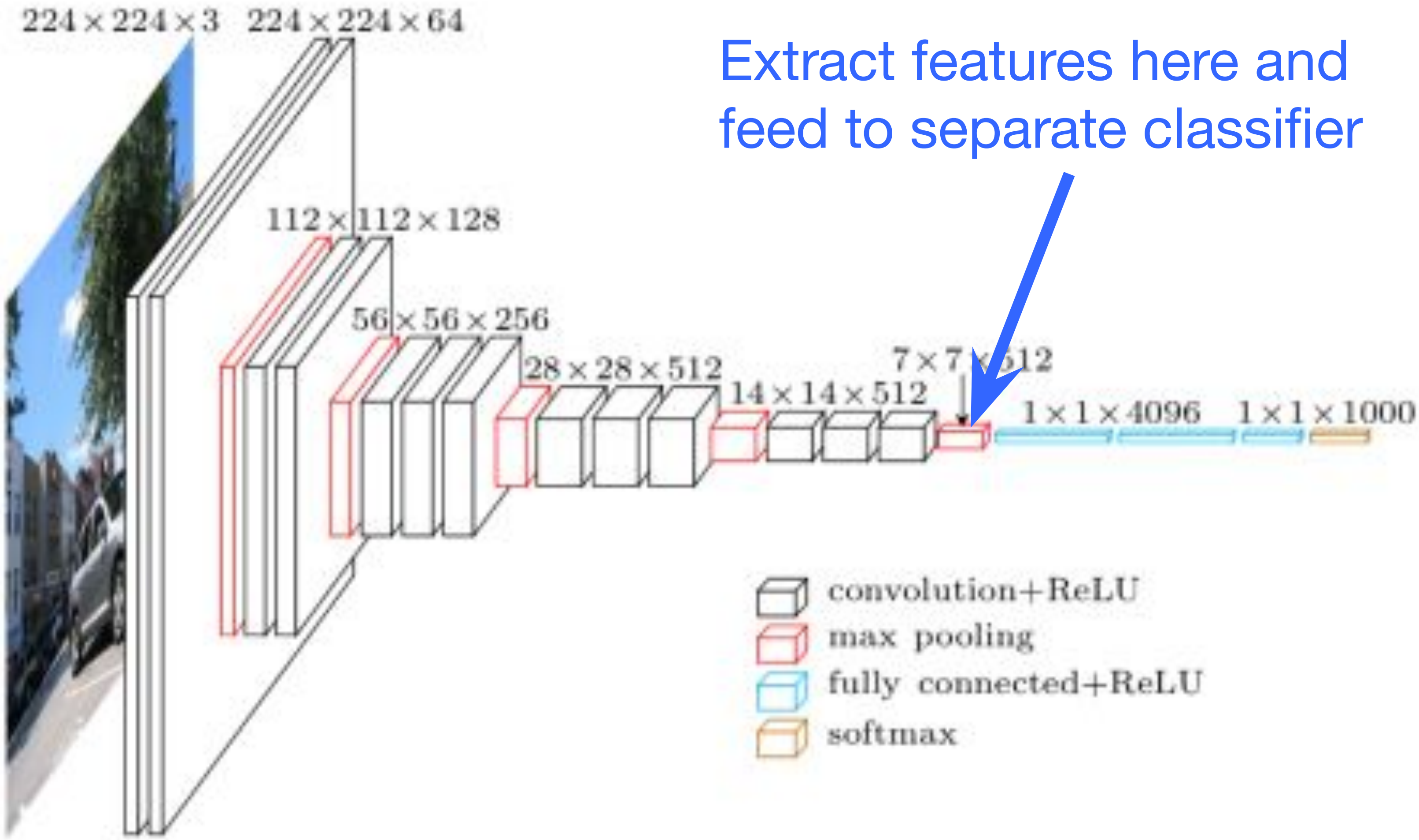
Lee et al. 'Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations' ICML 2009

PRE-TRAINED MODEL



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

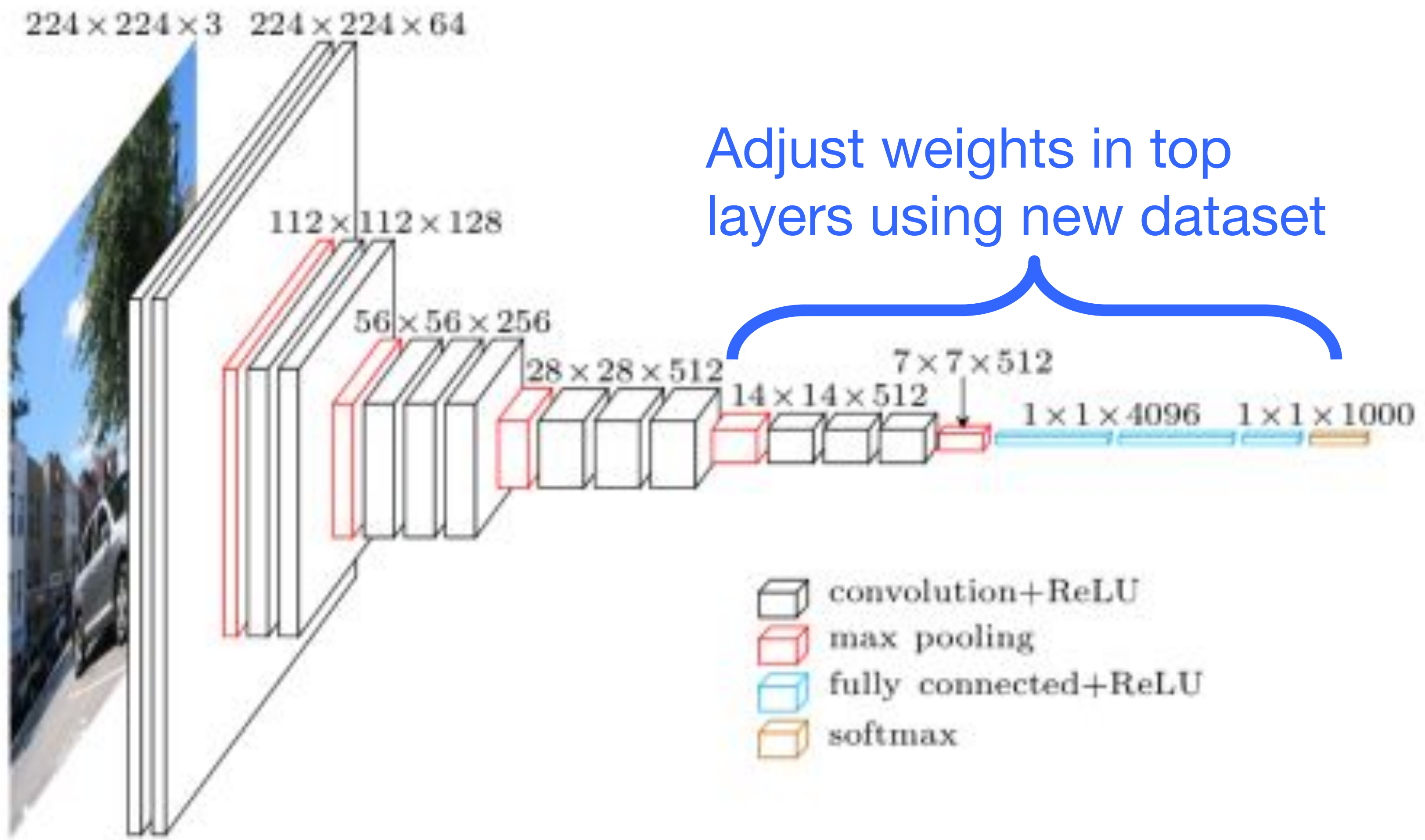
TRANSFER LEARNING - FEATURE EXTRACTION



Extract features here and feed to separate classifier

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

TRANSFER LEARNING - FINE TUNING



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

TRANSFER LEARNING EXERCISES

- **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

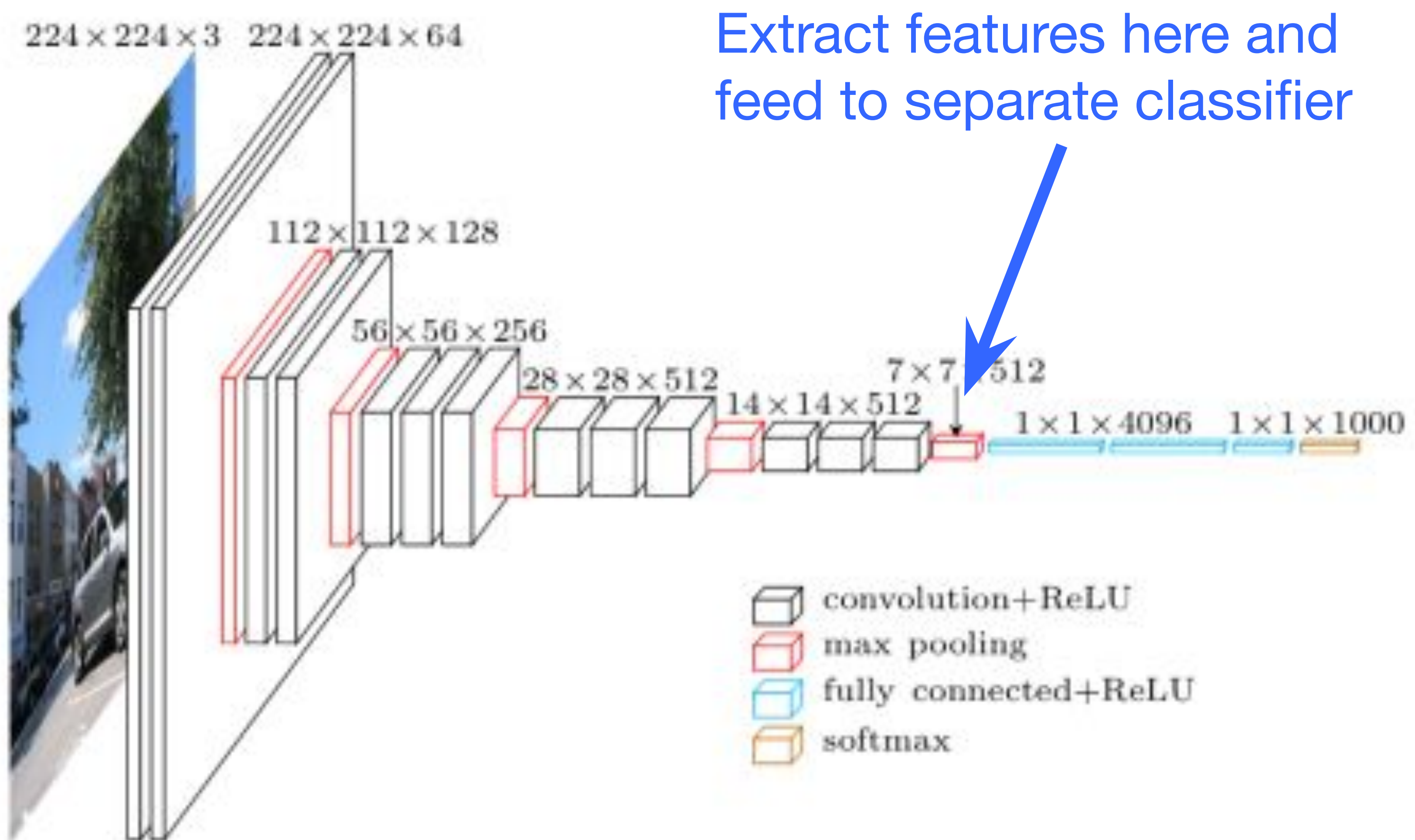
- **Subset of Kaggle cats and dogs dataset**
- **Train**
  - 1000 cats + 1000 dogs
- **Validation**
  - 400 cats + 400 dogs
- **Test**
  - 400 cats + 400 dogs

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



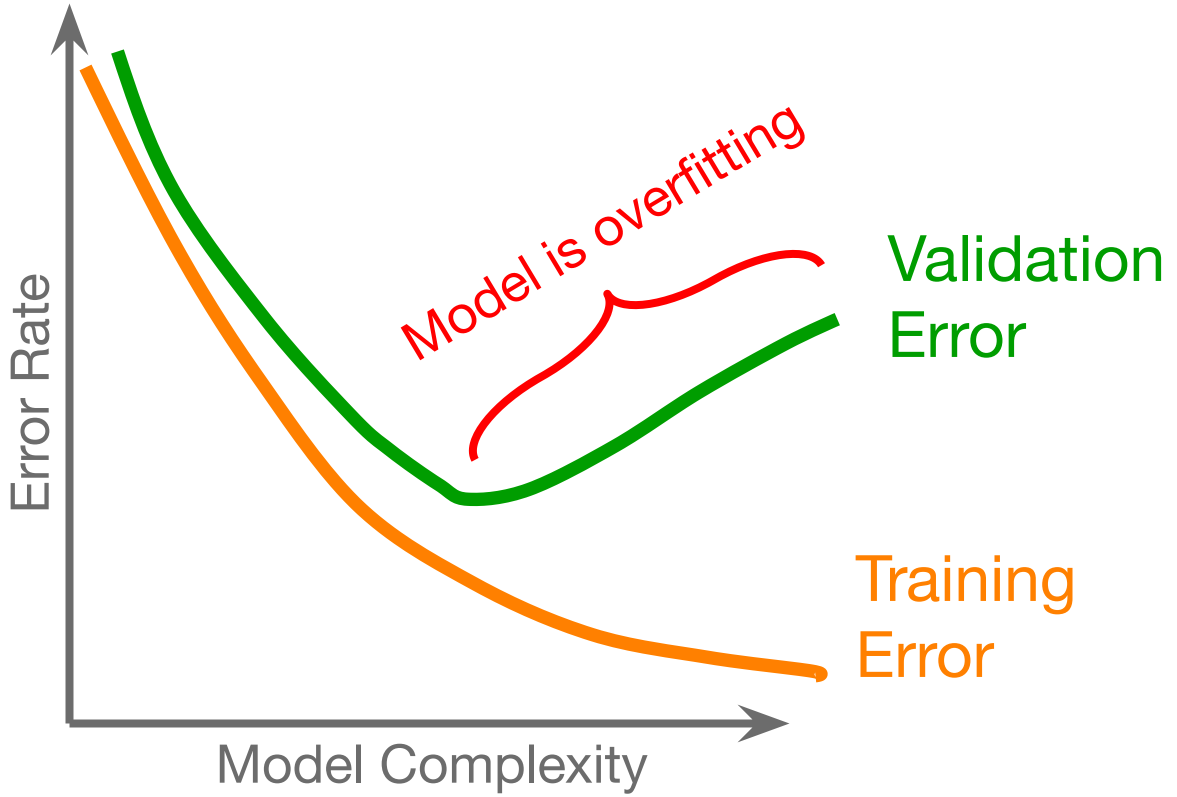
Source: https://www.cs.toronto.edu/~frossard/post/vgg16/

Feature Extraction Overview

- **Data**
  - Set image dimensions & location
  - 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 model weights
  - Use validation data to determine when to stop training
- **Evaluate model**
  - Calculate accuracy, etc.
  - Perform inference on test images

Early Stopping

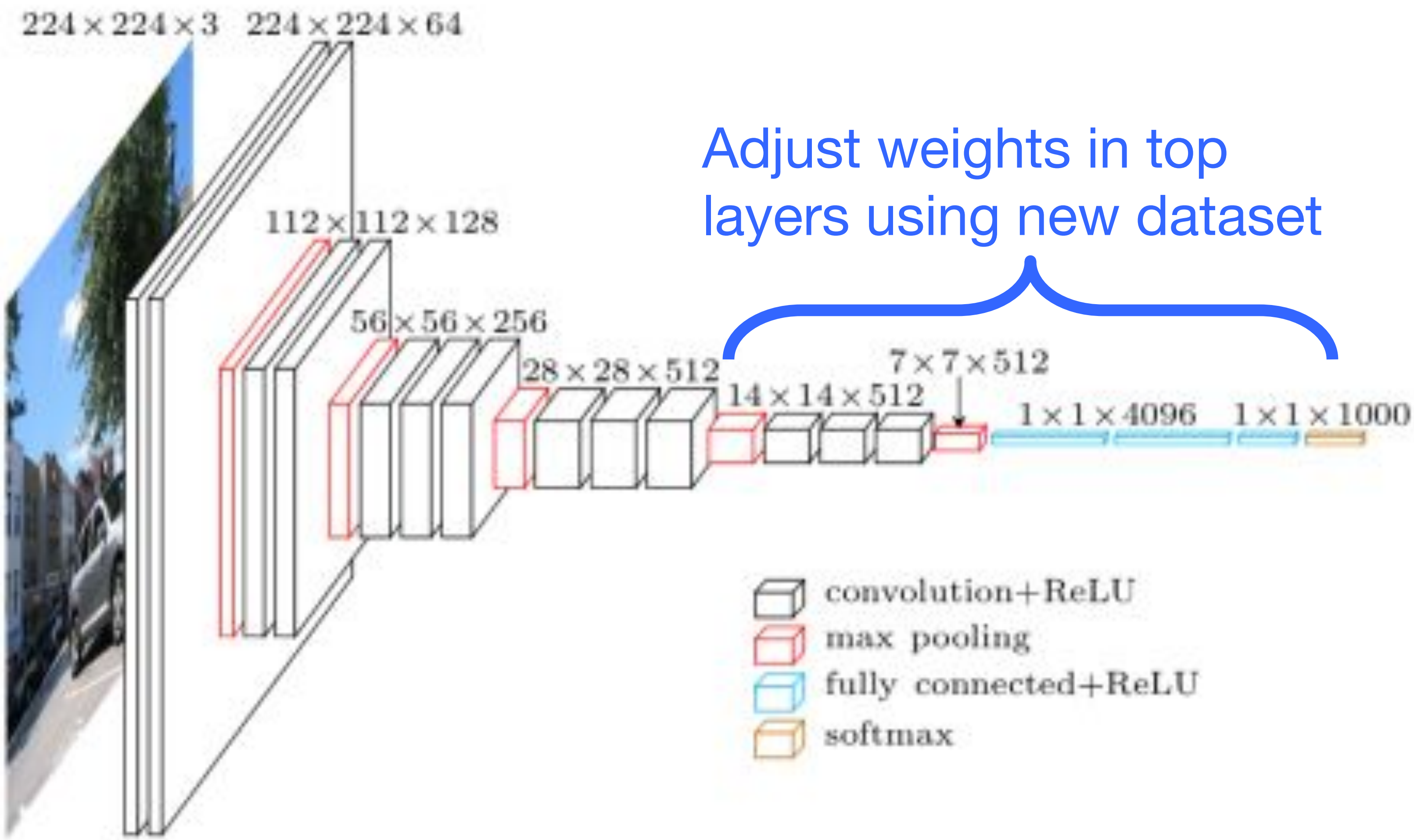
Using validation data to determine when to stop training to avoid overfitting



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



Source: 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/

OTHER PRACTICAL TIPS

- **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

Fine Tune Overview

- **Data**
  - Set image dimensions & location
  - Use ImageDataGenerator to read images from folder
- **Model**
  - Load trained model from feature extraction code
  - Freeze weights up to last convolutional block
  - Weights in last convolutional block and top classifier will be adjusted during training
- **Train model**
  - Use training data to adjust model weights
  - Use validation data to determine when to stop training
- **Evaluate model**
  - Calculate accuracy, etc.
  - Perform inference on test images

Code

- **features_extract_tf.ipynb**
  - Transfer learning with feature extraction

- **finetune_tf.ipynb**
  - Transfer learning with fine tuning

- **Note**
  - Close features_extract_tf.ipynb before running finetune_tf.ipynb to avoid out-of-memory errors

Setup

- **Login to Expanse**
  - Open terminal window on local machine
  - ssh login.expanse.sdsc.edu
- **Pull latest from repo**
  - git pull
  - URL: https://github.com/ciml-org/ciml-summer-institute-2021

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**
  - start_gpu
    - Alias for: galyleo.sh launch --account ‘sds184’ --reservation ‘ciml-day3’ --partition ‘gpu-shared’ --cpus-per-task 10 --memory-per-node 93 --gpus 1 --time-limit 03:00:00 --jupyter ‘lab’ --notebook-dir “/home/${USER}” --env-modules ‘singularitypro’ --sif ‘/cm/shared/apps/containers/singularity/tensorflow/tensorflow-latest.sif’ --bind ‘/expanse,/scratch,/cvmfs’ --nv --quiet

- **To check queue**
  - squeue -u $USER

REFERENCES

- **TensorFlow tutorial**
  - https://github.com/tensorflow/docs/blob/master/site/en/tutorials/images/transfer_learning.ipynb
- **TensorFlow/Keras API**
  - https://www.tensorflow.org/api_docs/python/tf/keras/Model
- **Transfer Learning**
  - http://cs231n.github.io/transfer-learning/