Code Time:
Deep Learning with TensorFlow, Keras and PyTorch

Topics Taken

May 9. Paper review due on the same day:

- Nick Fioravanti, Justin Serwinski; Jalen Winfield, Deep Learning in Neural Networks with Neural Data Analysis: EEG

- Alexander Wang, A.I. Philosophy and Ethical Alignment

May 16. Paper review due on the same day:

- Josh Miller; Marc Mounzer; Kevin Chavez, Kasonde Chewe, Computational Neuroscience

- Christopher Campbell: Application in Biomedicine

- Eleni Alexakis, Hadis Jami, Tony Okeke, Medical Imaging

Roadmap

- Walk-Through
  - ✓ A walk-through with code for using TensorFlow (Keras) step-by-step

  [`tensorflow4prediction_demo.ipynb – Keras implementation, Tensorboard callback, & Excercise`]

- Image classification using transfer learning based on pre-trained model

  [dog/cat: `pre_trained_EfficientNet.ipynb`]

Deep Learning with TensorFlow, Keras and fastai

The fastai library simplifies training fast and accurate neural nets using modern best practices

TensorFlow: https://www.tensorflow.org/

Keras: https://keras.io/

Fastai: https://www.fast.ai/

TensorFlow

https://www.tensorflow.org/

What is TensorFlow?

Open source software library for
numerical computation using data flow graphs

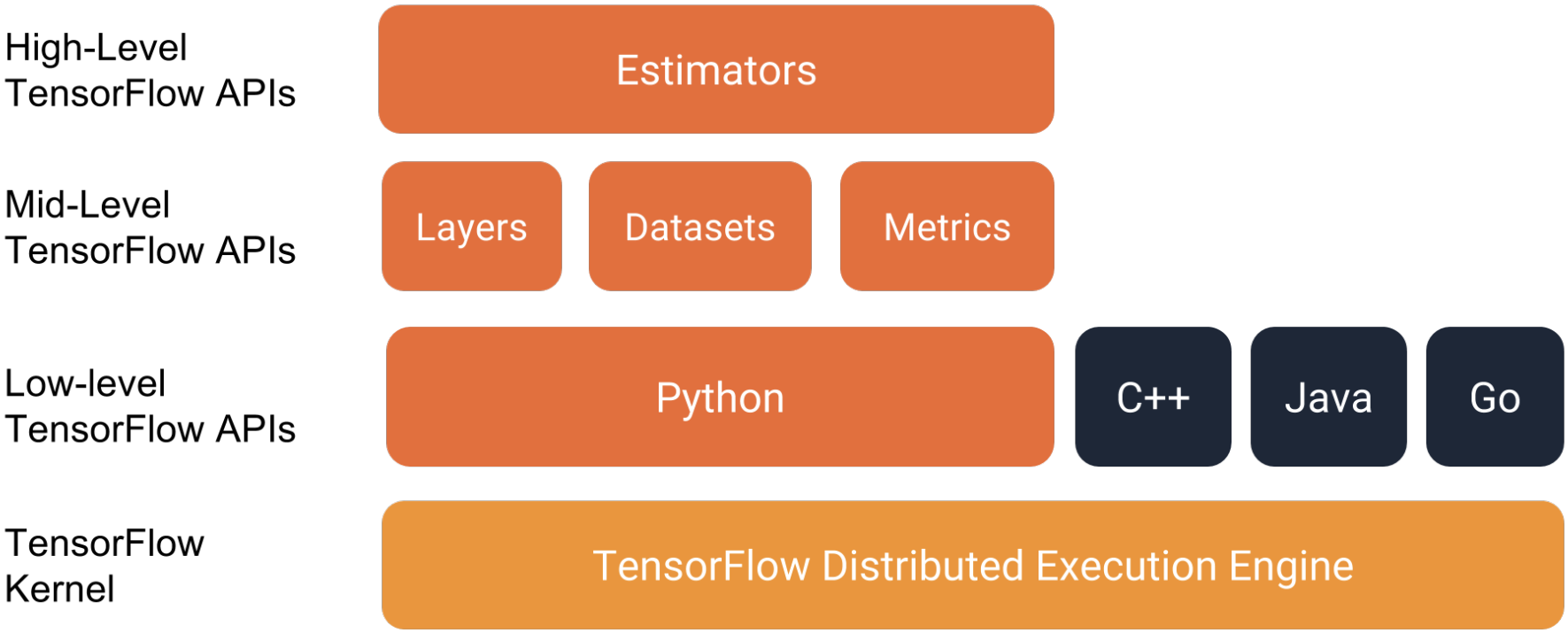
Why TensorFlow?

Flexibility + Scalability + Popularity

Originally developed by Google as a single infrastructure for machine learning in both production and research

Who are using TensorFlow?

TensorFlow: Programming Environment



https://www.tensorflow.org/

TensorFlow

Three main components:

- The TensorFlow API, written in C++, contains the API to define the models and train the models with data. It also has a user-friendly Python interface.
- TensorBoard is a visualization toolkit to help with analyzing, visualizing, and debugging TensorFlow graphs
- TensorFlow Serving is a flexible, high-performance serving system used to deploy pre-trained machine learning models in production. Also written in C++ and accessible with a Python interface, Serving is able to switch from old to new models instantaneously

TensorFlow has been used widely in academic research and industrial applications. Some notable current uses include Deep Speech, RankBrain, SmartReply, and On-Device Computer Vision.

Models, tutorials of TensorFlow at this GitHub repo.

When to use TensorFlow?

- Experimenting with new machine learning architectures and approaches
- Operationalizing your experiments
- Iterating on various architectures
- Large-scale distributed models
- Building models for mobile / embedded systems

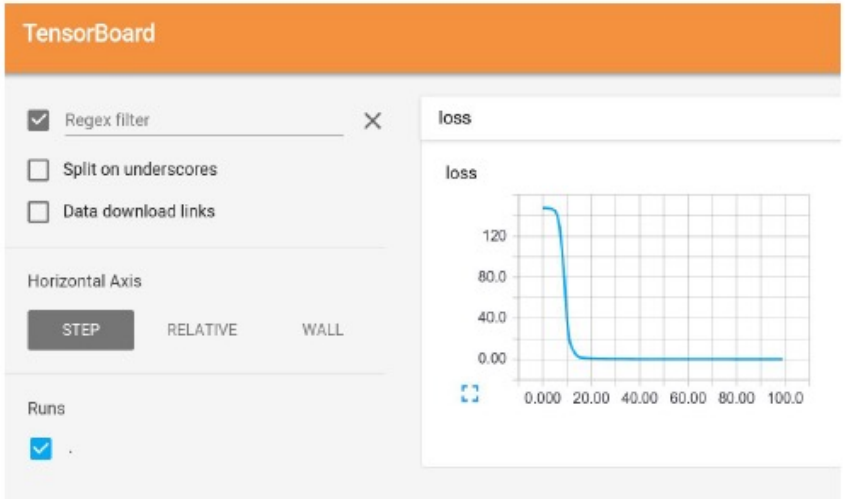
What is a Tensor?

TensorFlow uses a tensor data structure to represent all data. A TensorFlow tensor as an n-dimensional array or list. A tensor has a static type, a rank, and a shape.

- An n-dimensional array
- 0-d tensor: scalar (number)
  5 (Rank 0)
- 1-d tensor: vector:
  [1., 2., 3.] (Rank 1, Shape [3])
- 2-d tensor: matrix:
  [[1., 2., 3.], [4., 5., 6.]] (Rank 2, Shape [2, 3])

TensorFlow: Tensorboard

Add logging to code to record loss, stats, etc
Run server and get pretty graphs!



Try TensorFlow

https://www.tensorflow.org/tutorials

- Start by cloning the TensorFlow models repo from GitHub

```
>> git clone https://github.com/tensorflow/models
```

Why Keras?

Many Python-based ML:

- scikit-learn provides an easy way to run many classical ML algorithms, such as linear regression and support vector machines.
- At the other end of the spectrum are PyTorch and Google's TensorFlow, which give you greater control over the inner workings of your deep learning model.
- Microsoft's CNTK and Berkeley's Caffe are similar deep learning frameworks, which have Python APIs to access their C++ engines.

Keras:

- A wrapper around TensorFlow and CNTK, with Amazon's MXNet coming soon. (It also works with Theano, but the University of Montreal stopped working on this in September 2017.)
- It provides an easy-to-use API for building models that you can train on one backend, and deploy on another.
- Another reason to use Keras, rather than directly using TensorFlow, is that Core ML includes a Keras converter, but not a TensorFlow converter
- State-of-the-art deep learning image classifiers in Keras: VGGNet, ResNet, Inception, and Xception pre-trained on the ImageNet dataset

Keras Code Time

Install prerequisites

Instructions for Mac or Linux. You need to install a recent version of Python (2.7 or 3), plus the packages keras, numpy, matplotlib and jupyter.

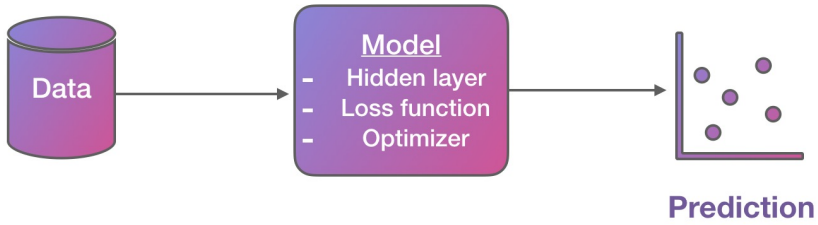
We'll go over the very high-level ideas in class. If you're new to computer programming, Python, or Jupyter notebooks, you'll want to run through some tutorials to help you get started.

Introduction to Python (beginner)
Optional: Introduction to Git (beginner) and GitHub tutorial.
Video introduction to Jupyter Notebooks (with code) (beginner)
Learning to code with Python *and* Jupyter Notebooks (beginner)
https://www.tensorflow.org/install/

Useful Python packages:

Introduction to Pandas (intermediate)
Practice your Pandas skills (intermediate)
Getting started with Numpy (intermediate)
Getting started with Scipy (intermediate)

A walk-through with code for using TensorFlow.keras



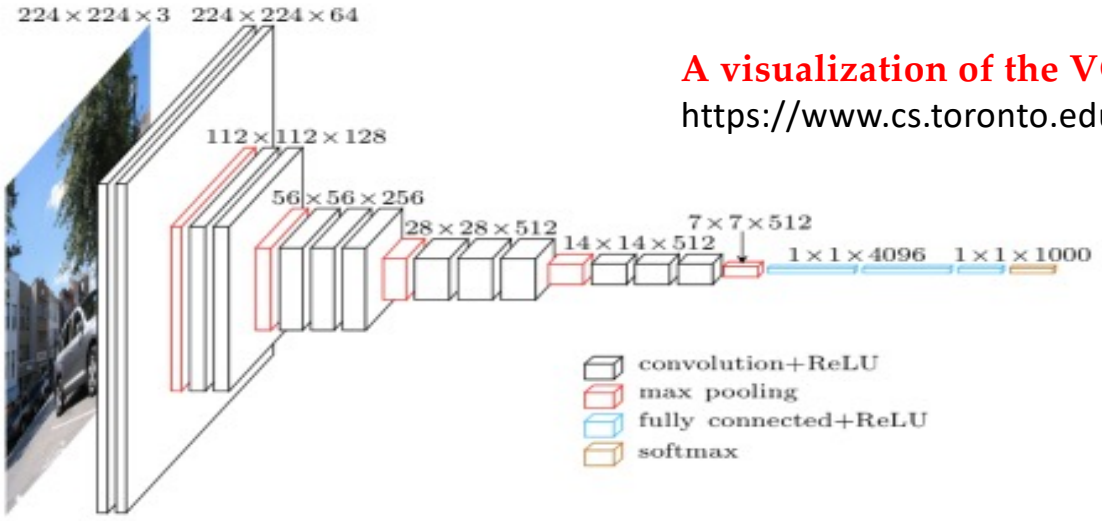
~~Demo: Tensorflow_Intro.ipynb (1.x)~~

tensorflow4prediction_demo.ipynb – Keras implementation

Transfer Learning

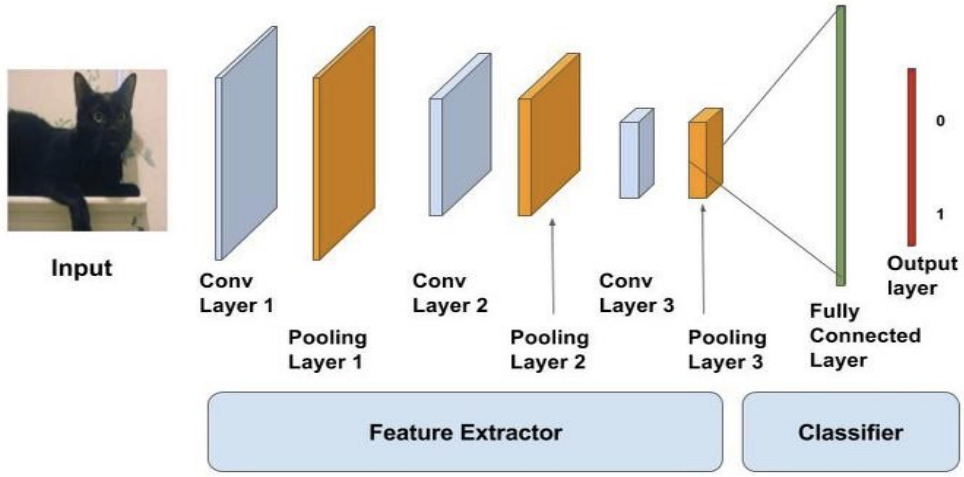
Transfer Learning: Why?

- More data → better learning, but
  - Huge data required: difficult to get such huge labeled datasets for training
  - Huge computing power required: Even with data, a large amount of time (100s hours) to train the network
- Leverage the models already trained on very large amounts of data for difficult tasks with thousands of classes: AlexNet, VGGNet, Inception, ResNet, Xception, MobileNet and many more! See: https://github.com/keras-team/keras/tree/master/keras/applications



A visualization of the VGG architecture
https://www.cs.toronto.edu/~frossard/post/vgg16/

Transfer Learning: How?

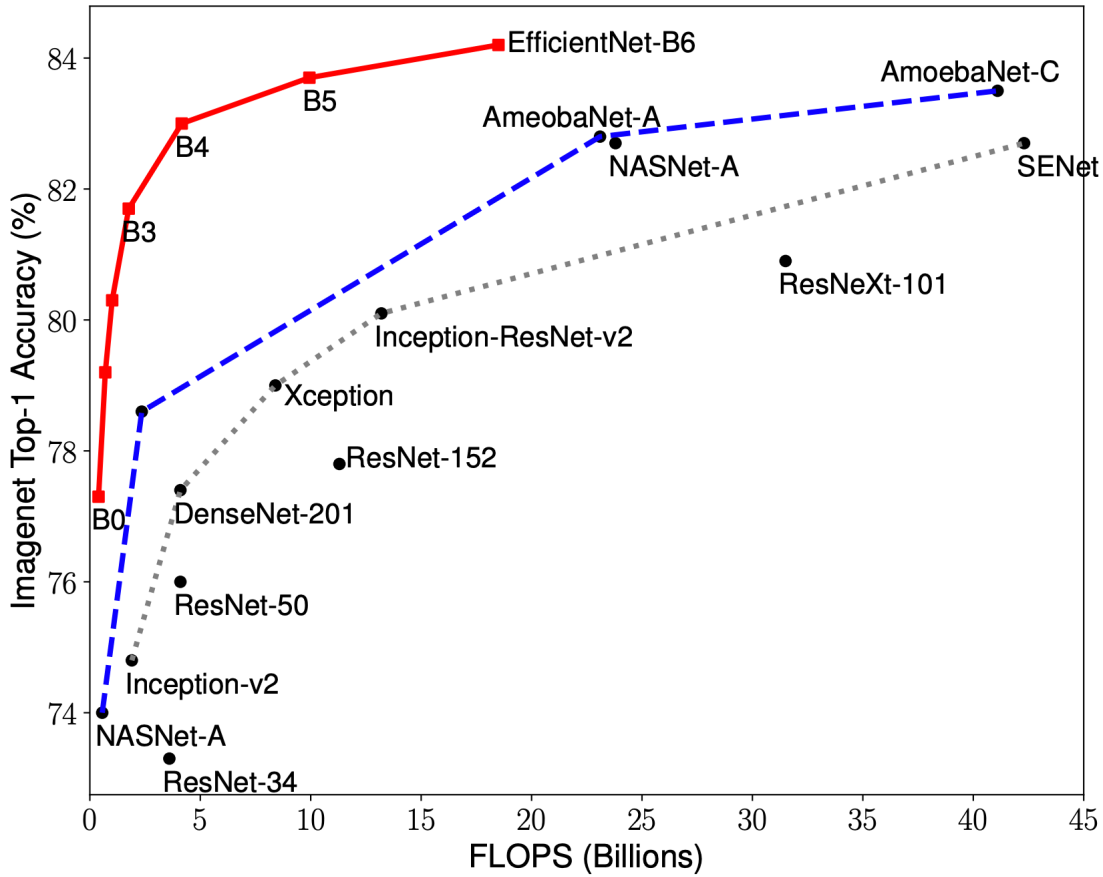


- **Pretrained models:** Image classification directly using different pre-trained models
- **Transfer Learning:** Training a classifier for a different task, using the features extracted from the pretrained models. Freeze conv-layers (Features) + Linear Classifier (eg. SVM)
- **Fine-tuning:** Training a classifier for a different task, by modifying the weights of the above models. Tweak all the layers of the ConvNet or keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portion.

Transfer Learning: a rule of thumb

- New dataset is small and similar to original dataset, use transfer learning
- New dataset is large, use fine-tuning, either all the layers or only some higher-level portion of the network.
- Earlier features of the pretrained models contain more generic features (e.g. edge detectors or color blob detectors) that should be useful to many tasks, but later layers more specific to the details of the classes contained in the original dataset.
- VGGNet etc quite large (over 533MB for VGG16 and 574MB for VGG19). To build apps, smaller network architectures are often more desirable (such as SqueezeNet, GoogLeNet, etc.)

Accuracy vs. efficiency



Pre-Trained Models for Image Classification

- VGG-16
- ResNet50
- Inceptionv3
- EfficientNet

- EfficientNetV2
- NF-Nets (Normalizer Free Nets): fast a

Tan and Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, arXiv:1905.11946, 2020

Transfer Learning Demo

`pre_trained_EfficientNet.ipynb`

Best ML Models 2021 for each Area

Image Classification: EfficientNetV2, NF-Nets
Image Segmentation: Efficient U-Nets
Object Detection: YoloV5, VarifocalNet (VF-Net)
Tabular Data & Time series: LGBM, XGBoost and Catboost, GNN
NLP: GPT-3, BERT

Dogs vs. Cats | Kaggle

- The training archive contains 25,000 images: dogs and cats
- The test dataset has 10000 unlabelled images.
- Use a much smaller dataset to demonstrate Pre-Trained models for image classification.

https://www.kaggle.com/c/dogs-vs-cats/data

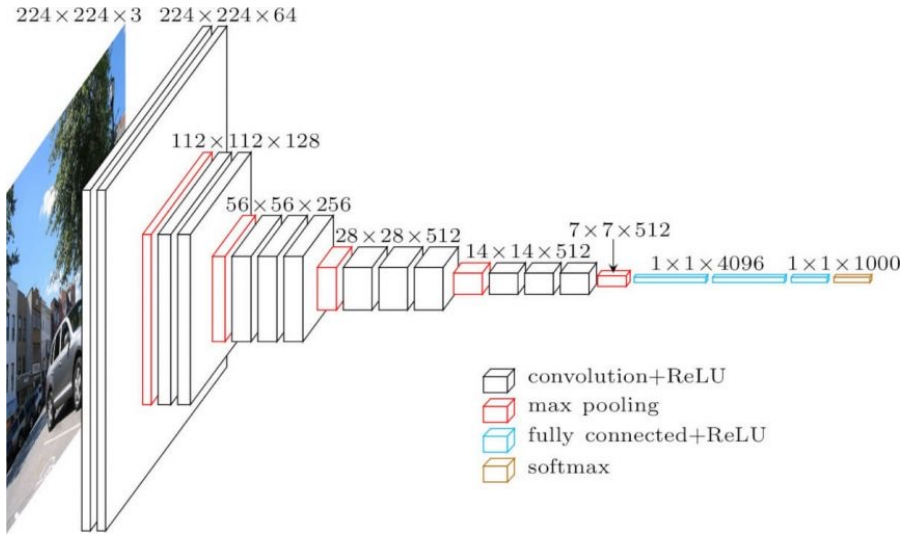
Transfer learning allows us to use a pre-existing model, trained on a huge dataset, for our own tasks

- Setting up the system
- Preparing the Dataset
- Pre-Trained Models for Image Classification
    - ✓ VGG-16
    - ✓ ResNet50
    - ✓ Inceptionv3
    - ✓ EfficientNet
- Deployment: Image Classification in Tensorflow / Keras using Gradio (demo)

Top 4 Pre-Trained Models

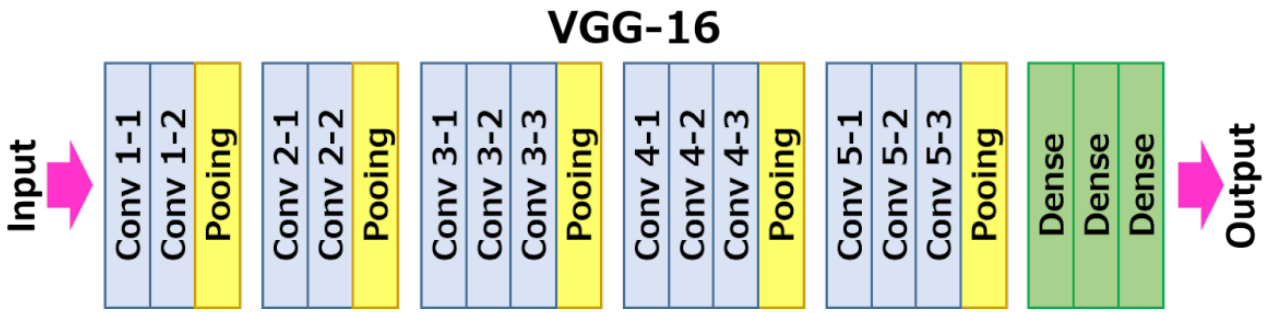
Model	Year	Number of Parameters	Top-1 Accuracy
VGG-16	2014	138 Million	74.5%
ResNet-50	2015	25 Million	77.15%
Inception V3	2015	24 Million	78.8%
EfficientNetB0	2019	5.3 Million	76.3%
EfficientNetB7	2019	66 Million	84.4%

VGG-16

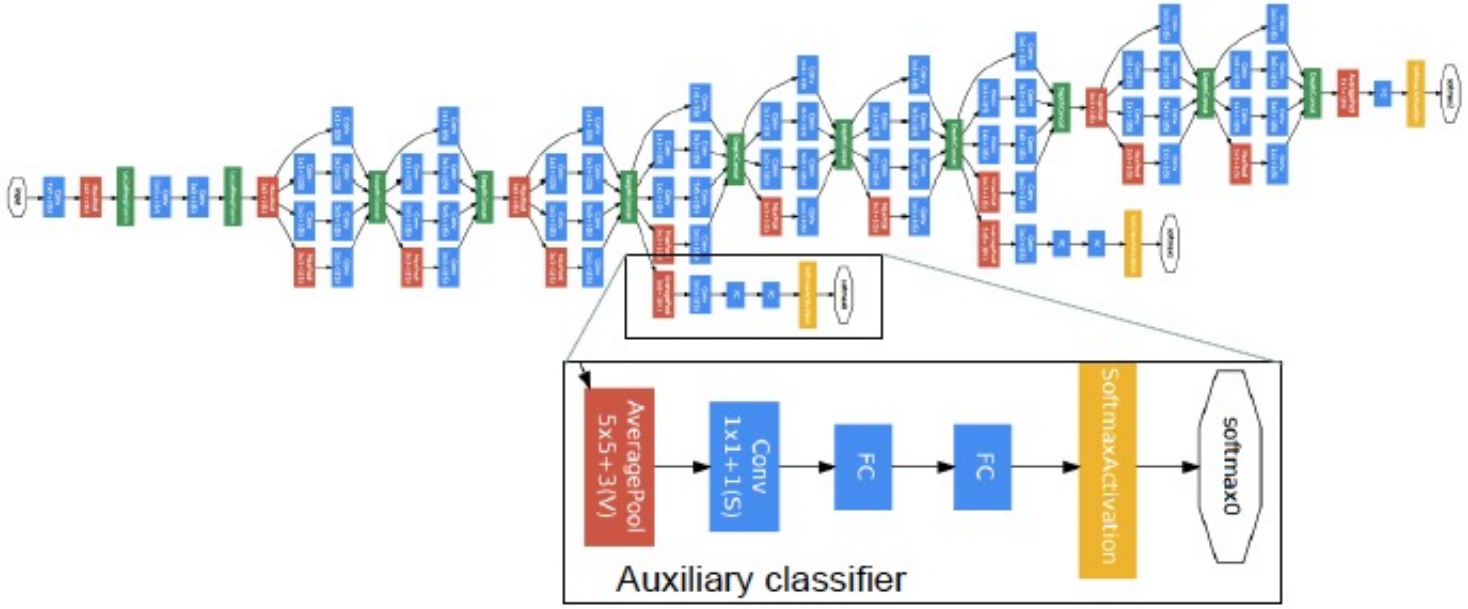


The layers of the model:

- Convolutional Layers = 13
- Pooling Layers = 5
- Dense Layers = 3

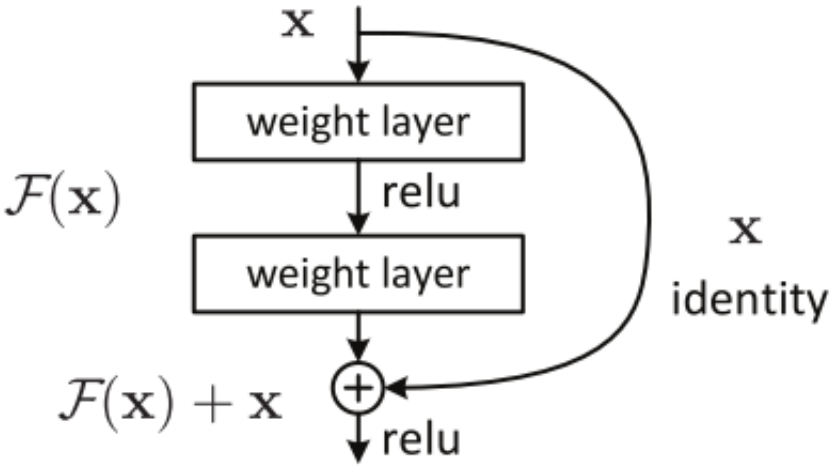


Inception or GoogLeNet (22-layer)



C. Szegedy et al., Going deeper with convolutions, CVPR 2015

ResNet: ILSVRC 2015 winner – 152 layers



Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

EfficientNet

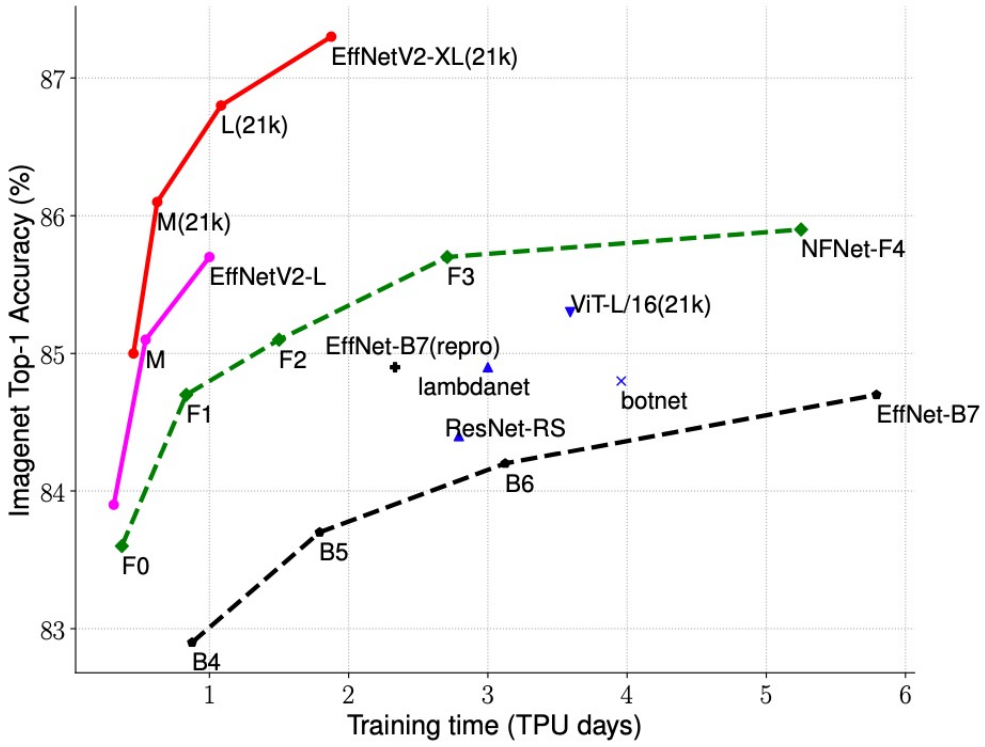
Baseline B0 model - EfficientB0 has only 5.3 million parameters!

State of the Art ML Models in 2021

- **Image Classification**
  - ✓ EfficientNetsV2: 2% better in performance while training 5–11x times faster
  - ✓ Normalizer-Free Networks (NF-Nets): **remove batch normalization** and introduce Adaptive Gradient Clipping to a 9x boost in training speed whilst having the same SOTA performance on ImageNet
- **Image Segmentation**
  - ✓ Efficient U-Nets
- **Object Detection**
  - ✓ YoloV5
  - ✓ VarifocalNet (VF-Net)
- **Tabular data & Time-series**
  - ✓ Gradient Boosting Machines (CatBoost > Light GBM > XGBoost)
  - ✓ Graph Neural Networks (GNNs)

Tan and Le, EfficientNetV2: Smaller Models and Faster Training, 2021
Brock et al. High-Performance Large-Scale Image Recognition Without Normalization, 2021

EfficientNetsV2



Tan and Le, EfficientNetV2: Smaller Models and Faster Training, 2021

Big Tech & Their Favourite Deep Learning Techniques