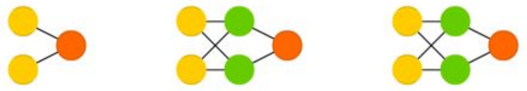
Transfer learning

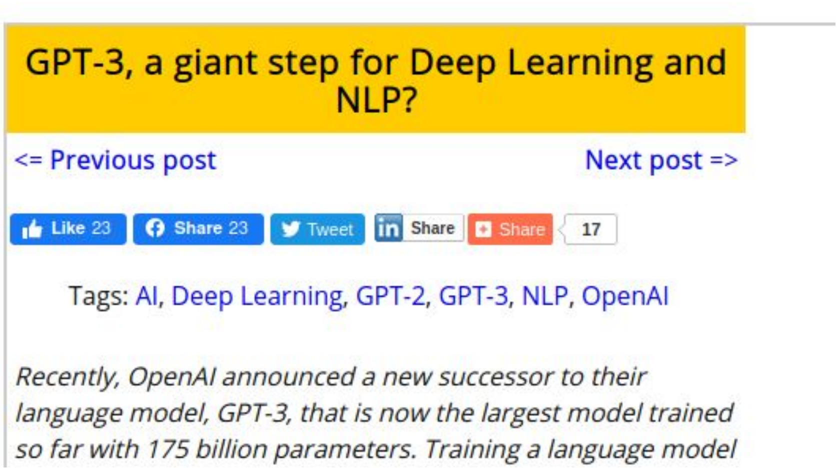
Let other people do the legwork

Filippo Biscarini
Senior Scientist
CNR, Milan (Italy)

Nelson Nazzicari
Senior Scientist
CREA, Lodi (Italy)



Deep Learning headlines are intimidating



GPT-3, a giant step for Deep Learning and NLP?

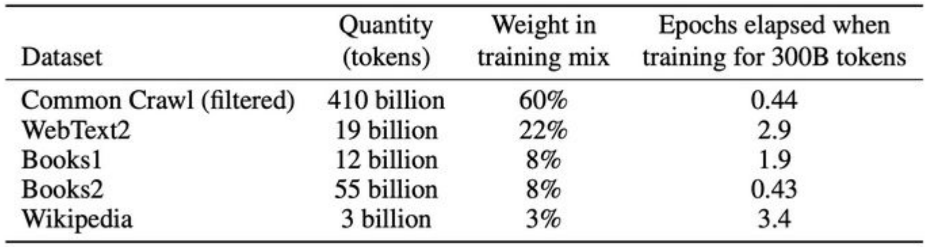
<= Previous post   Next post =>

Like 23 | Share 23 | Tweet | Share | Share | 17

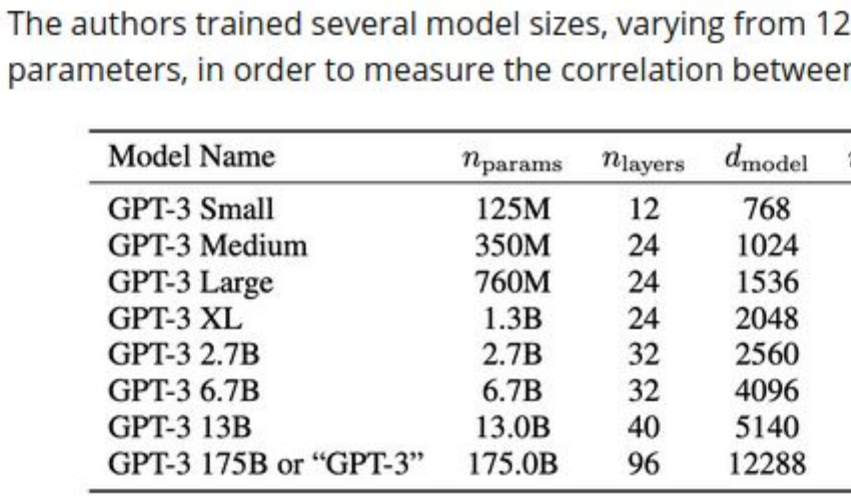
Tags: AI, Deep Learning, GPT-2, GPT-3, NLP, OpenAI

Recently, OpenAI announced a new successor to their language model, GPT-3, that is now the largest model trained so far with 175 billion parameters. Training a language model

- They added known high-quality corpora to the training mix.



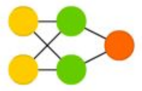
Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4



The authors trained several model sizes, varying from 12
parameters, in order to measure the correlation betweer

Model Name	n_{params}	n_{layers}	d_{model}
GPT-3 Small	125M	12	768
GPT-3 Medium	350M	24	1024
GPT-3 Large	760M	24	1536
GPT-3 XL	1.3B	24	2048
GPT-3 2.7B	2.7B	32	2560
GPT-3 6.7B	6.7B	32	4096
GPT-3 13B	13.0B	40	5140
GPT-3 175B or "GPT-3"	175.0B	96	12288

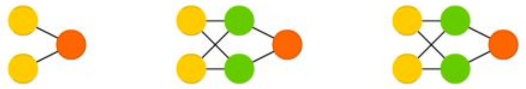
https://www.kdnuggets.com/2020/06/gpt-3-deep-learning-nlp.html



What is transfer learning?

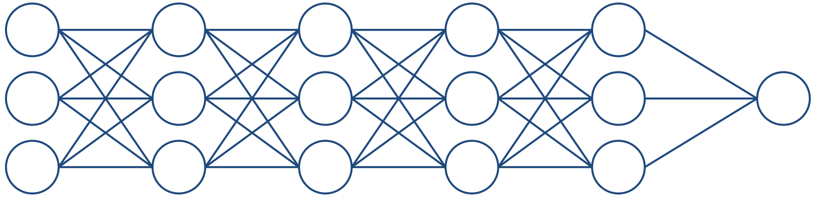
"In transfer learning, we first train a base network on a base dataset and task, and then we repurpose the learned features, or *transfer* them, to a second target network to be trained on a target dataset and task."

- How transferable are features in deep neural networks?, Yosinski et al., 2014



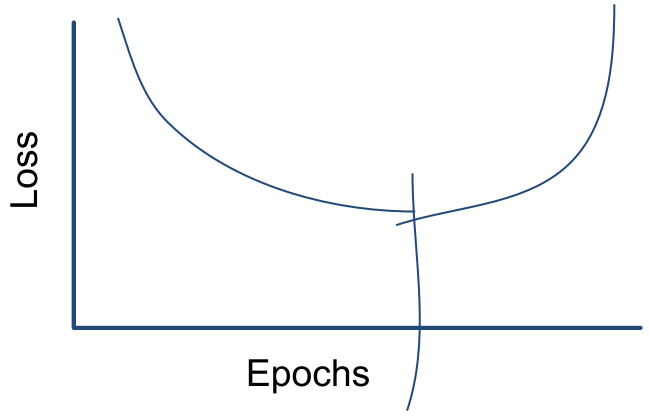
Transfer learning workflow

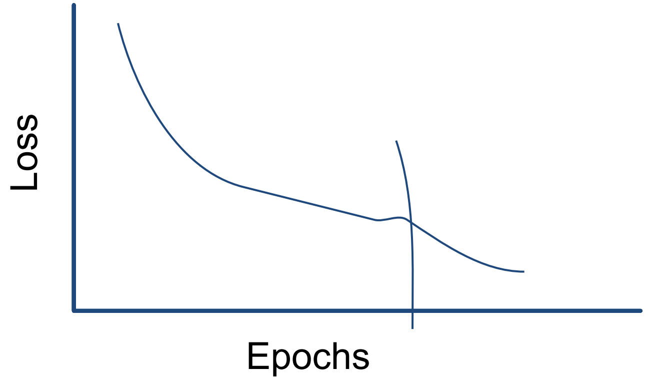
1. Get architecture
2. Get parameters
3. Remove last layer
4. Freeze
5. A little wrapper
6. Train
7. <optional> Fine tune

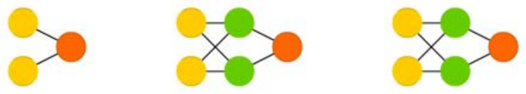


What is fine tuning IN TRANSFER LEARNING

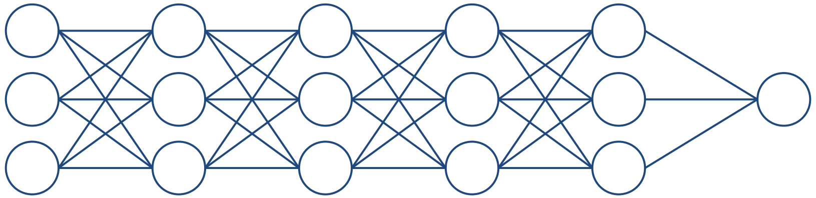
1. <once you have trained the top layers>
2. Unfreeze everything
3. Select a very small learning rate
   a. (10+ smaller than before)
4. Do some training epochs



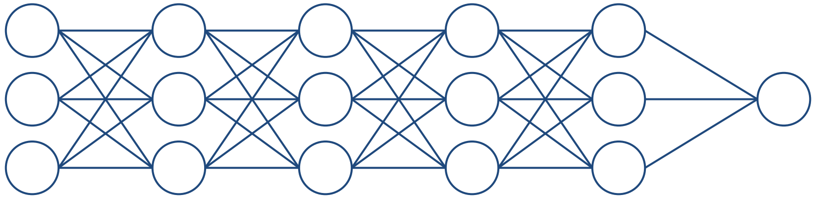




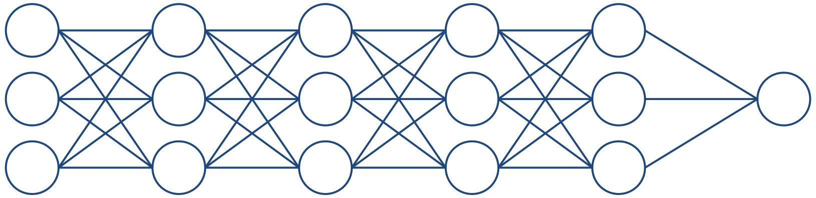
You can train more!



100-1000 samples
Only last layer is trained, everything else is frozen



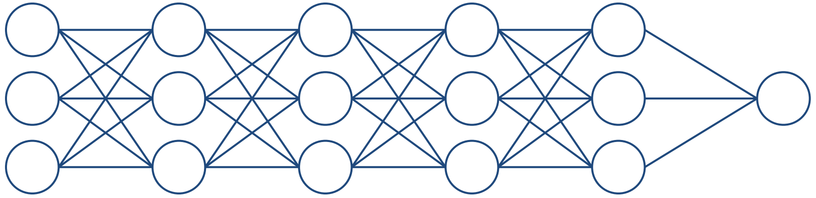
1000-10000 samples
Unfreeze the last 1-5 layers



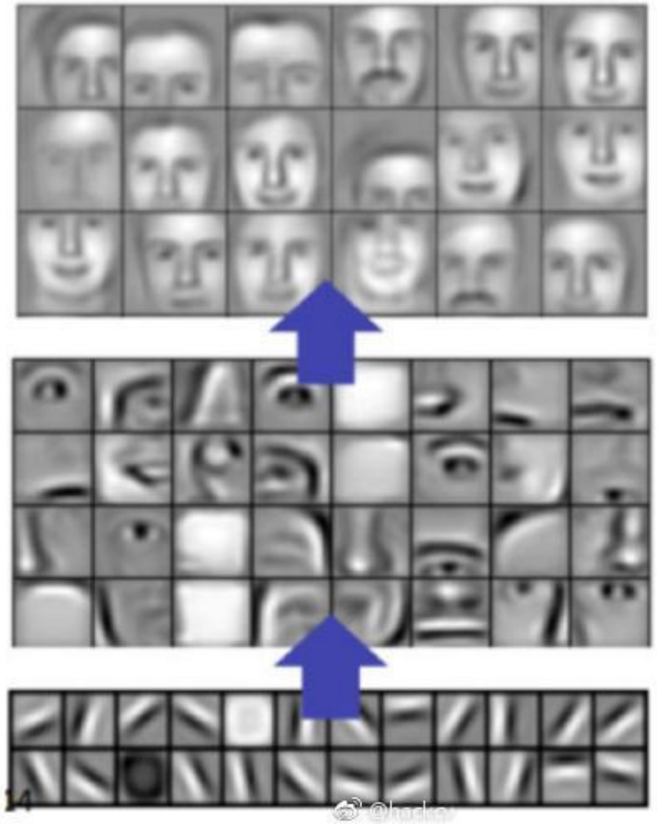
50000-100000 samples
Unfreeze everything

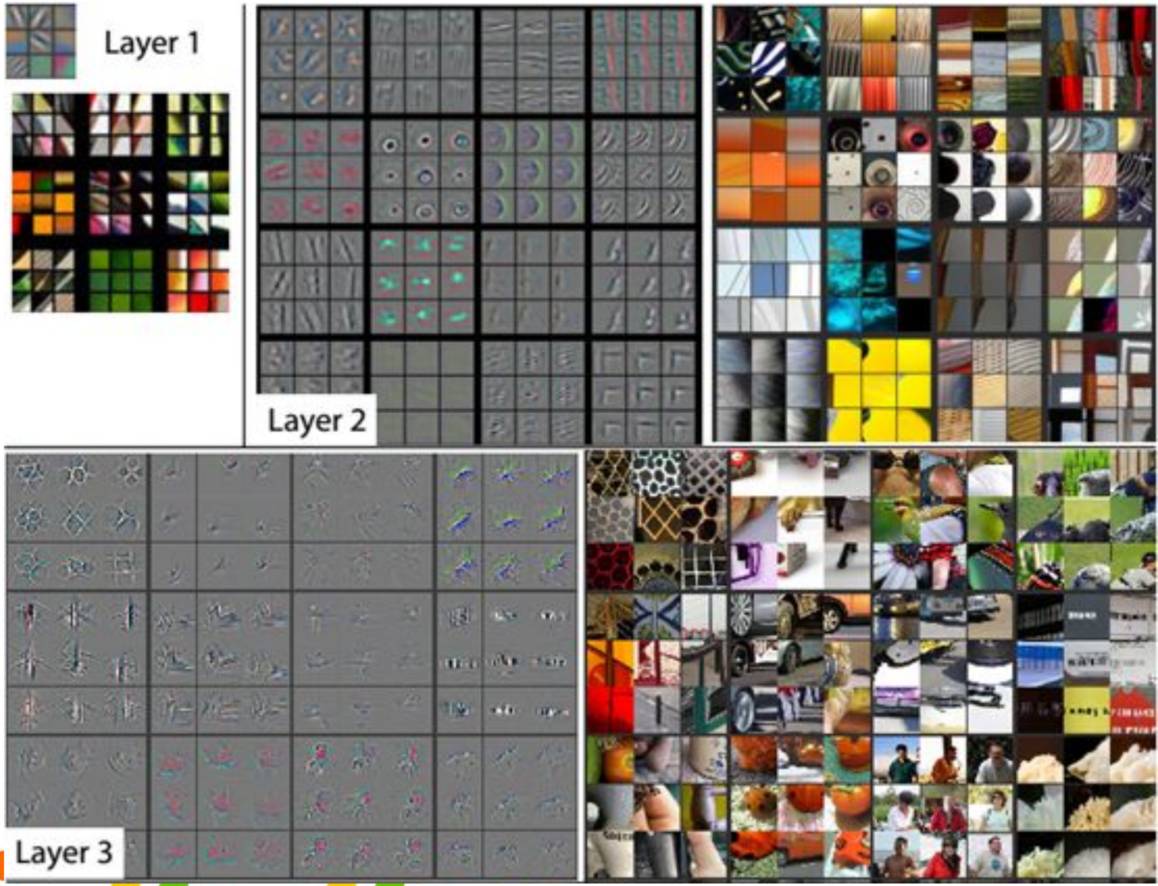
A different approach: feature extraction

1. Get architecture
2. Get parameters
3. Remove last layer
4. Pass all your data through the network
5. Throw away the network
6. Do a small network

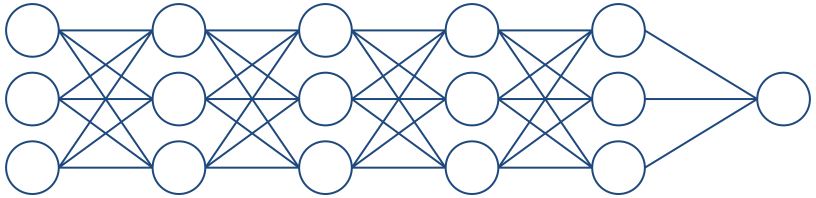


Why does it work?

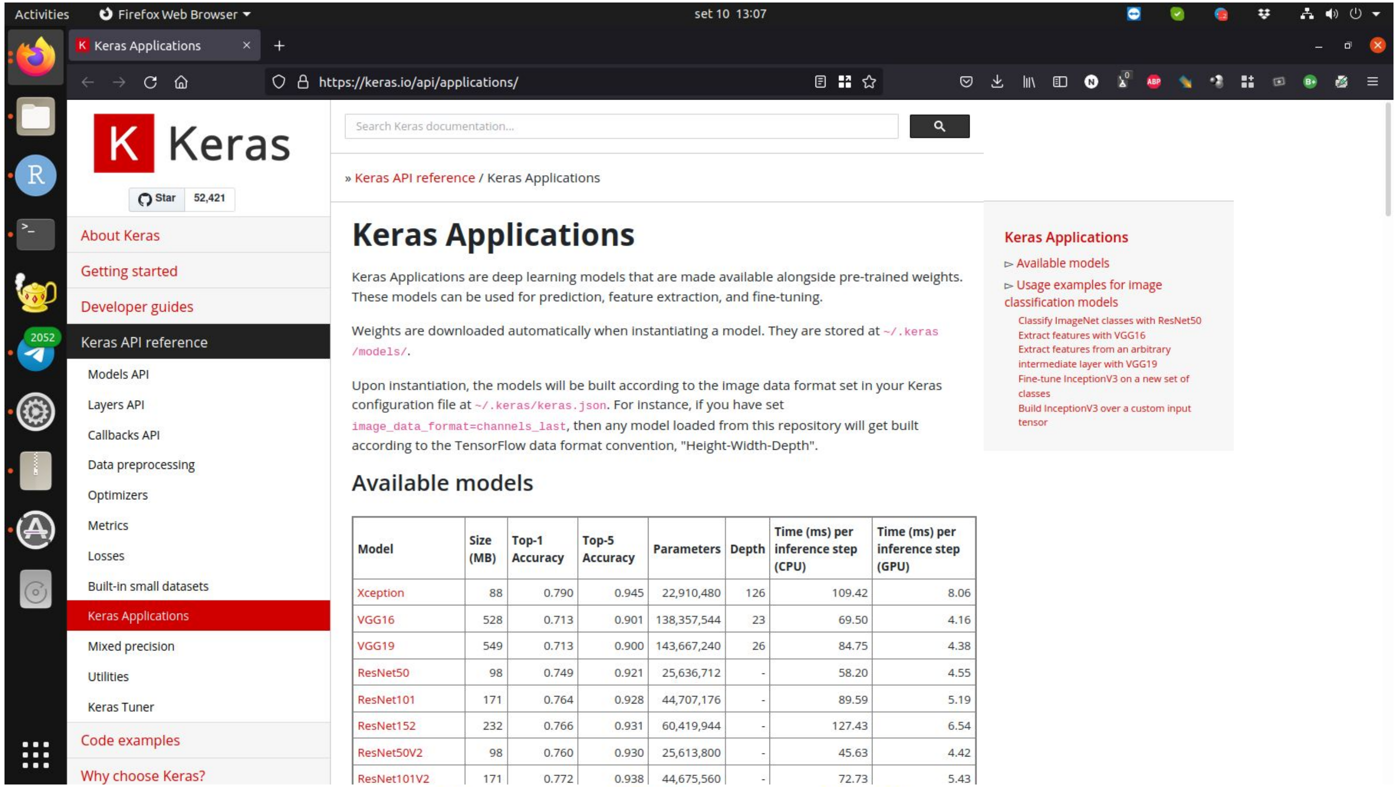




Why does it work?



In keras: available architectures (“applications”) https://keras.io/api/applications/



About Keras
Getting started
Developer guides
Keras API reference
- Models API
- Layers API
- Callbacks API
- Data preprocessing
- Optimizers
- Metrics
- Losses
- Built-in small datasets
- Keras Applications
- Mixed precision
- Utilities
- Keras Tuner

Code examples
Why choose Keras?

» Keras API reference / Keras Applications

Keras Applications

Keras Applications are deep learning models that are made available alongside pre-trained weights. These models can be used for prediction, feature extraction, and fine-tuning.

Weights are downloaded automatically when instantiating a model. They are stored at `~/.keras/models/`.

Upon instantiation, the models will be built according to the image data format set in your Keras configuration file at `~/.keras/keras.json`. For instance, if you have set `image_data_format=channels_last`, then any model loaded from this repository will get built according to the TensorFlow data format convention, "Height-Width-Depth".

Available models

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	0.790	0.945	22,910,480	126	109.42	8.06
VGG16	528	0.713	0.901	138,357,544	23	69.50	4.16
VGG19	549	0.713	0.900	143,667,240	26	84.75	4.38
ResNet50	98	0.749	0.921	25,636,712	-	58.20	4.55
ResNet101	171	0.764	0.928	44,707,176	-	89.59	5.19
ResNet152	232	0.766	0.931	60,419,944	-	127.43	6.54
ResNet50V2	98	0.760	0.930	25,613,800	-	45.63	4.42
ResNet101V2	171	0.772	0.938	44,675,560	-	72.73	5.43

Keras Applications
- ▹ Available models
- ▹ Usage examples for image classification models
  - Classify ImageNet classes with ResNet50
  - Extract features with VGG16
  - Extract features from an arbitrary intermediate layer with VGG19
  - Fine-tune InceptionV3 on a new set of classes
  - Build InceptionV3 over a custom input tensor

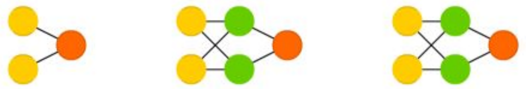
In keras

```python
from keras import models, layers
from keras.applications import ResNet50

#downloading the net and its weights trained on imagenet dataset
my_resnet = ResNet50(weights='imagenet', include_top=False, input_shape=(...))

#it's already the default value, but we set it anyway to
#make clear we are not going to train the whole thing
my_resnet.trainable = False

#build the model
model = models.Sequential()
model.add(my_resnet)
model.add(layers.Flatten())
model.add(layers.Dense(units=5, activation='softmax'))
```



[REF]

- How transferable are features in deep neural networks? https://arxiv.org/abs/1411.1792
- Language Models are Few-Shot Learners (GPT-3) https://arxiv.org/abs/2005.14165
- Deep Residual Learning for Image Recognition, He et al., 2015, https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html
- Rethinking the Inception Architecture for Computer Vision, Szegedy et al., 2016 https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.html
- EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, Tan & Le, 2019, https://arxiv.org/abs/1905.11946