Zurich University
of Applied Sciences

School of
Management and Law

Machine Learning II
“Classic” Deep Learning & Images – Transfer Learning and Fine Tuning

Dr. Elena Gavagnin
elena.gavagnin@zhaw.ch

FS2025

Machine Learning II – Images

Plan:

1. Artificial Neural Networks – Introduction, implementation with TF/Keras and Training

2. ”Classic” Deep Learning & Images – Convolutional Neural Networks and Transfer Learning

3. Multimodal and generative AI

”Classic” Deep Learning & Images (W8 and W9):

1. Working with visual data

2. What is a convolution?

3. Convolutional Neural Networks (CNN)

4. Famous CNN Architectures

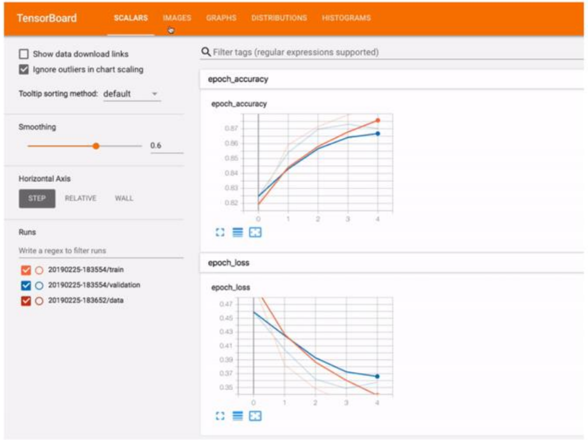
5. Reusing Pre-Trained Layers: Transfer Learning and Fine Tuning

6. Visual Data preparation

7. VisionTransformers and Self-Supervision

Recap– What happened in the last episode…

The Neural Network Real World: Tensorflow and Keras



Create a NN Model

```
model = tf.keras.models.Sequential([
tf.keras.layers.Dense(32, activation="relu", input_dim = 5),
tf.keras.layers.Dense(16, activation="relu"),
tf.keras.layers.Dense(10, activation="softmax")
])
```

Configure Model's losses & evaluation metrics

```
model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy
(from_logits=False),optimizer="sgd", metrics=["accuracy"])
```

Train the model

```
model.fit(X_train, y_train, epochs=15,
                    validation_data=(X_valid, y_valid))
```

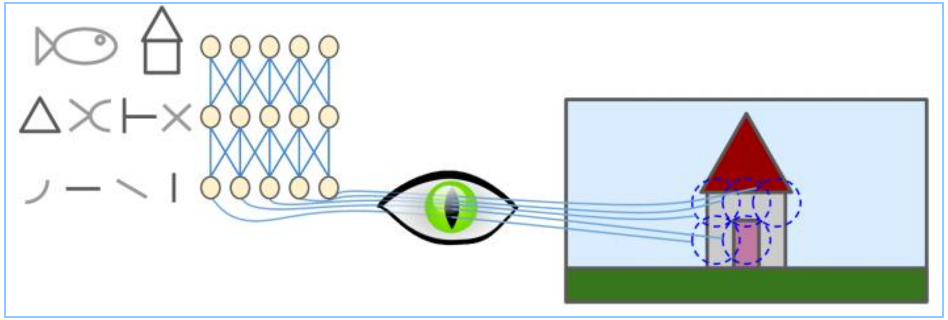
Evaluate the model

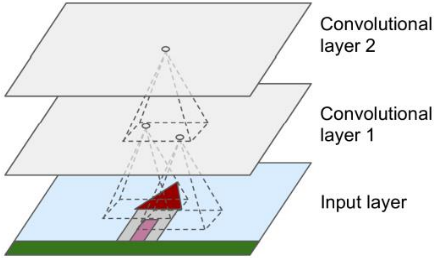
```
model.evaluate(X_test, y_test)
```

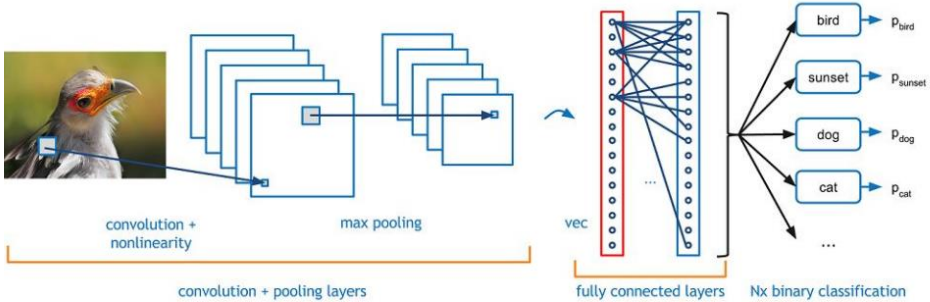

zh aw School of Management and Law

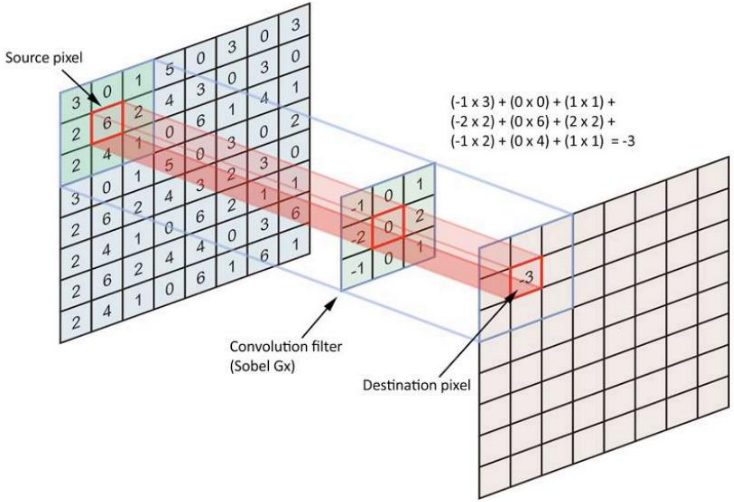
How do we do it..?

Take-Home Message:
Visual System as a Hierarchy of Feature Detectors









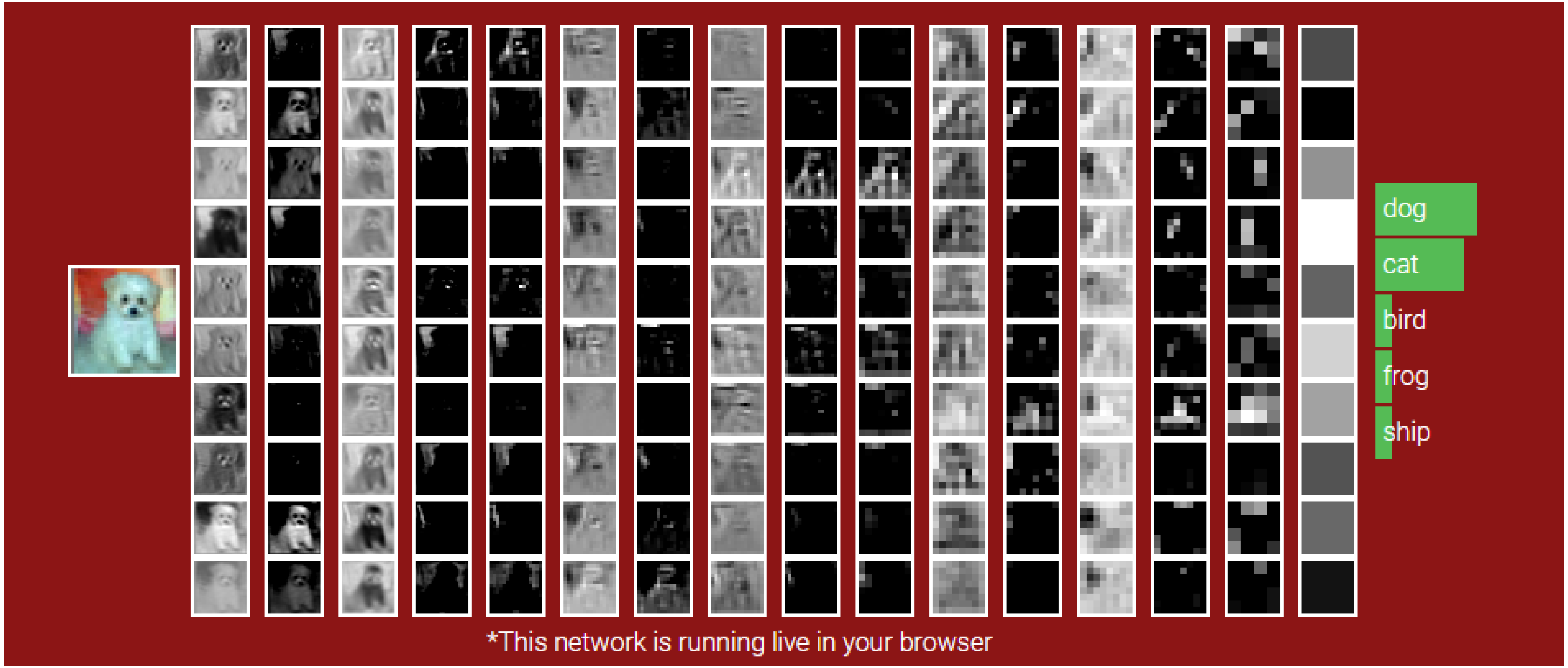
TF/Keras Implementation

So far, we flattened the images before feeding them as input to a NN. Now, CNN take a tensor of shape = [image_h, image_w, color_channels]

```
model2 = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation = 'softmax')
])
```

After you can attach a “classic” MLP/NN, with Flatten and Dense layers

Example in action



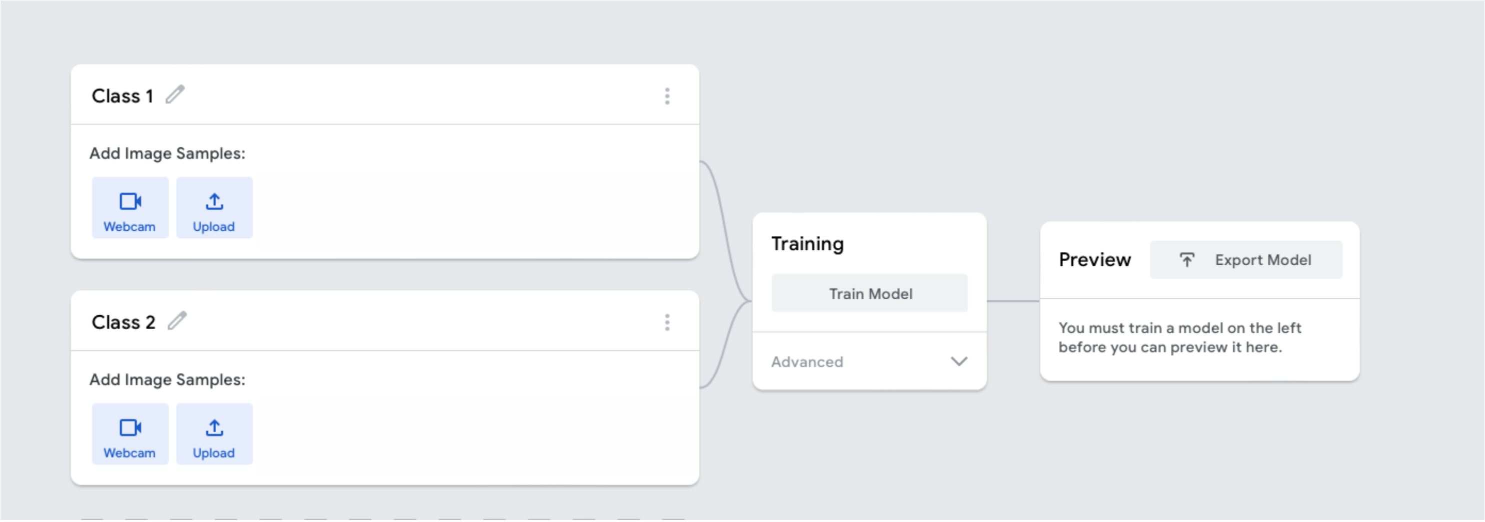
Stanford University CS231n: Convolutional Neural Networks for Visual Recognition

zh aw School of Management and Law

Motivational Warm-Up ☺

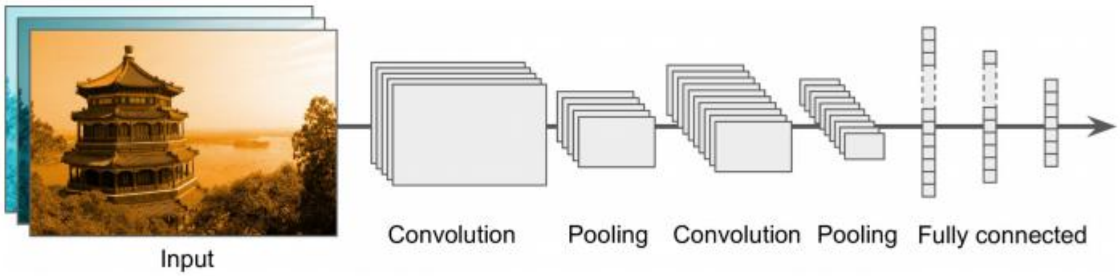
Interactive Exercise (2 People):

https://teachablemachine.withgoogle.com/train/image



Famous Architectures

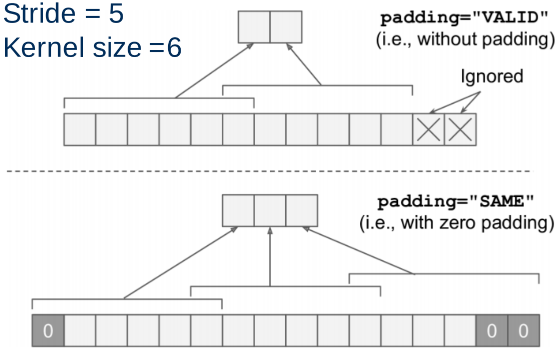
Famous Architectures



Do not use too big conv filters/kernel!
Better stacking 2 layers with filter 3x3 than 1 layer with 5x5. Why?
Less parameters, less overfit, lower computational cost
Remember! You apply the filter to all "channels"(color, input feature maps)

BUT it is ok to use a large (> 3x3) kernel in the input layer! Why?

Input = 13
Stride = 5
Kernel size =6



If stride =1 & padding = "same" -> input & output size are the same!

```
In [ ]: model = keras.models.Sequential([
    keras.layers.Conv2D(filters=64, kernel_size=7, input_shape=[28, 28, 1]),
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Conv2D(filters=128, kernel_size=3, activation='relu', padding="SAME"),
    keras.layers.Conv2D(filters=128, kernel_size=3, activation='relu', padding="SAME"),
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Conv2D(filters=256, kernel_size=3, activation='relu', padding="SAME"),
    keras.layers.Conv2D(filters=256, kernel_size=3, activation='relu', padding="SAME"),
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Flatten(),
    keras.layers.Dense(units=128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(units=64, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(units=10, activation='softmax'),
])
```

Nr filters increases!

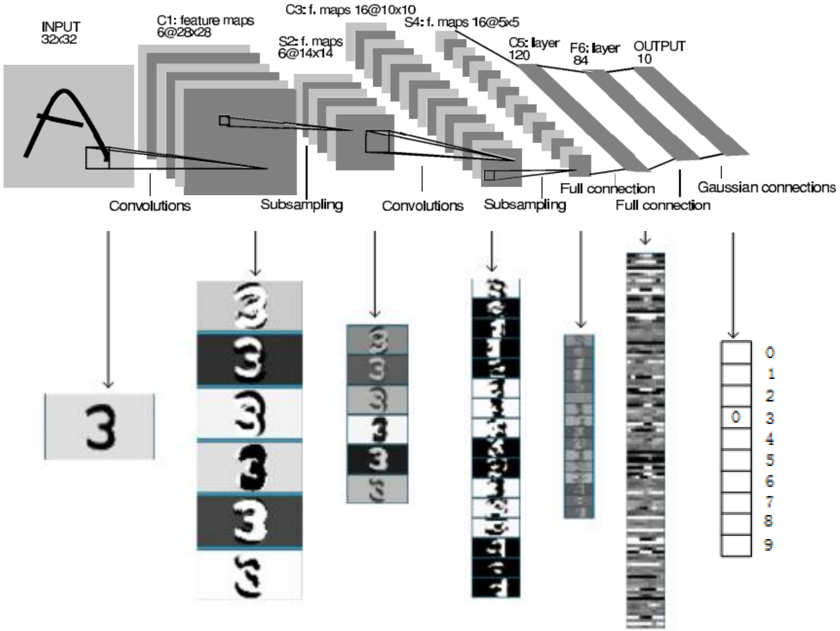
LeNet-5 (1998)

Yann LeCun

MNIST Data

http://yann.lecun.com/exdb/lenet/

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1 × 1	5 × 5	1	tanh
S4	Avg Pooling	16	5 × 5	2 × 2	2	tanh
C3	Convolution	16	10 × 10	5 × 5	1	tanh
S2	Avg Pooling	6	14 × 14	2 × 2	2	tanh
C1	Convolution	6	28 × 28	5 × 5	1	tanh
In	Input	1	32 × 32	–	–	–



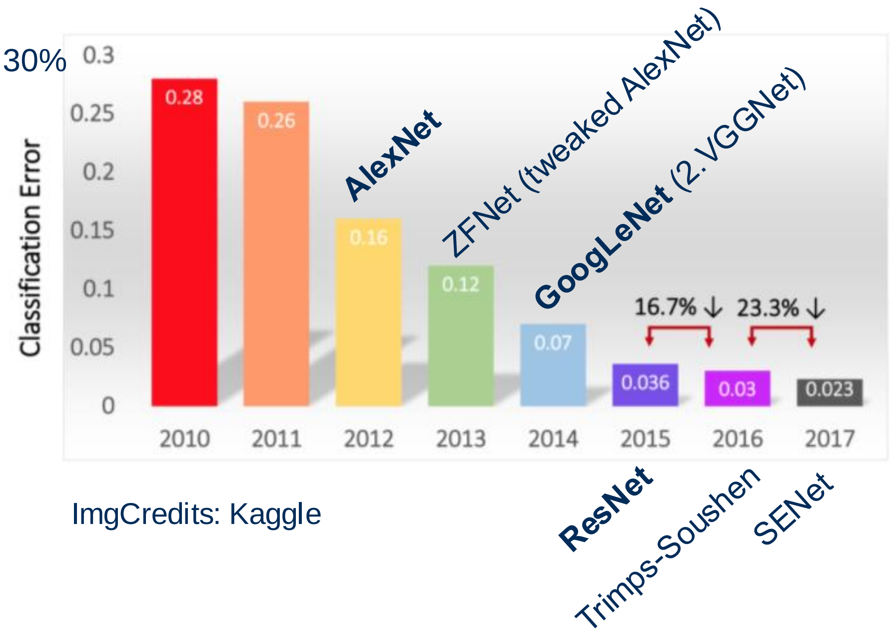
Origin. Paper: "Gradient-Based Learning Applied to Document Recognition", Y. LeCun, L. Bottou, Y. Bengio and P. Haffner (1998). http://www.iro.umontreal.ca/~lisa/bib/pub_subject/finance/pointeurs/lecun-98.pdf

Img credits: Orig Paper Y Le Cun + https://medium.com/analytics-vidhya/lenet-with-tensorflow-a35da0d503df

ILSVRC Winner Architectures

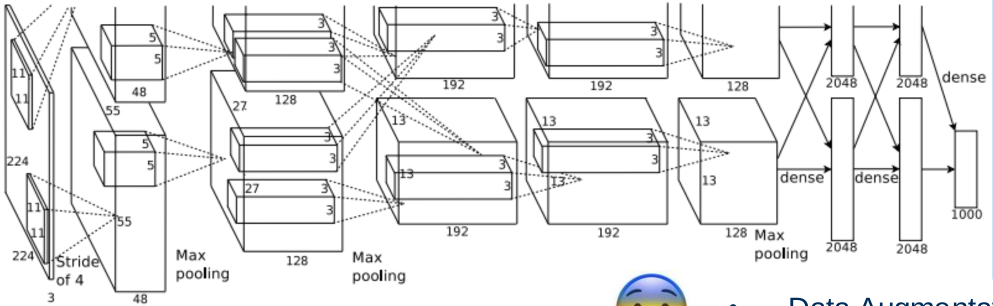
ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)

- Subset of ImageNet with roughly 1000 images in each of 1000 categories -> 1.2 million training images, 50,000 validation images, and 150,000 testing images
- Top-1 and top-5 error rates
- <u>Top-5 error rate</u> is the fraction of test images for which the correct label is <u>not</u> among the <u>5 labels considered most probable by the model</u>.
- ImageNet consists of variable-resolution images, need resizing



ImgCredits: Kaggle

AlexNet (2012)

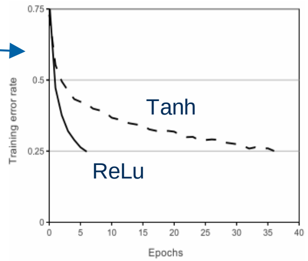


- top-5 error rate ~ 17% (second best 26%!)
- by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton.
- ~to LeNet-5 BUT
  - larger
  - Deeper
  - stacked convolutional layers directly on top of each other

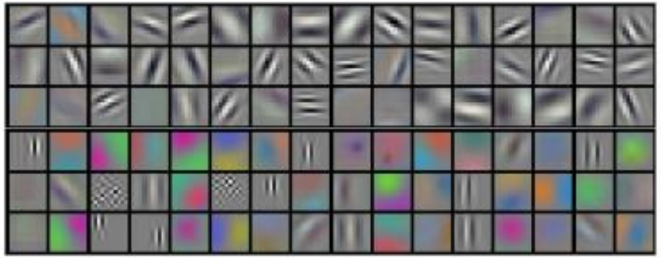
60 Million Parameters to learn!!!

Layer	Type	Maps	Size	Kernel size	Stride	Padding	Activation
Out	Fully Connected	–	1,000	–	–	–	Softmax
F9	Fully Connected	–	4,096	–	–	–	ReLU
F8	Fully Connected	–	4,096	–	–	–	ReLU
C7	Convolution	256	13 × 13	3 × 3	1	SAME	ReLU
C6	Convolution	384	13 × 13	3 × 3	1	SAME	ReLU
C5	Convolution	384	13 × 13	3 × 3	1	SAME	ReLU
S4	Max Pooling	256	13 × 13	3 × 3	2	VALID	–
C3	Convolution	256	27 × 27	5 × 5	1	SAME	ReLU
S2	Max Pooling	96	27 × 27	3 × 3	2	VALID	–
C1	Convolution	96	55 × 55	11 × 11	4	VALID	ReLU
In	Input	3 (RGB)	227 × 227	–	–	–	–

- Data Augmentation
- Dropout in FC



96 convolutional kernels of size 11×11×3 learned by the first convolutional layer



Orig. Paper, "ImageNet Classification with Deep Convolutional Neural Networks," A. Krizhevsky et al. (2012). https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

AlexNet (2012)

Most prob. 5 labels

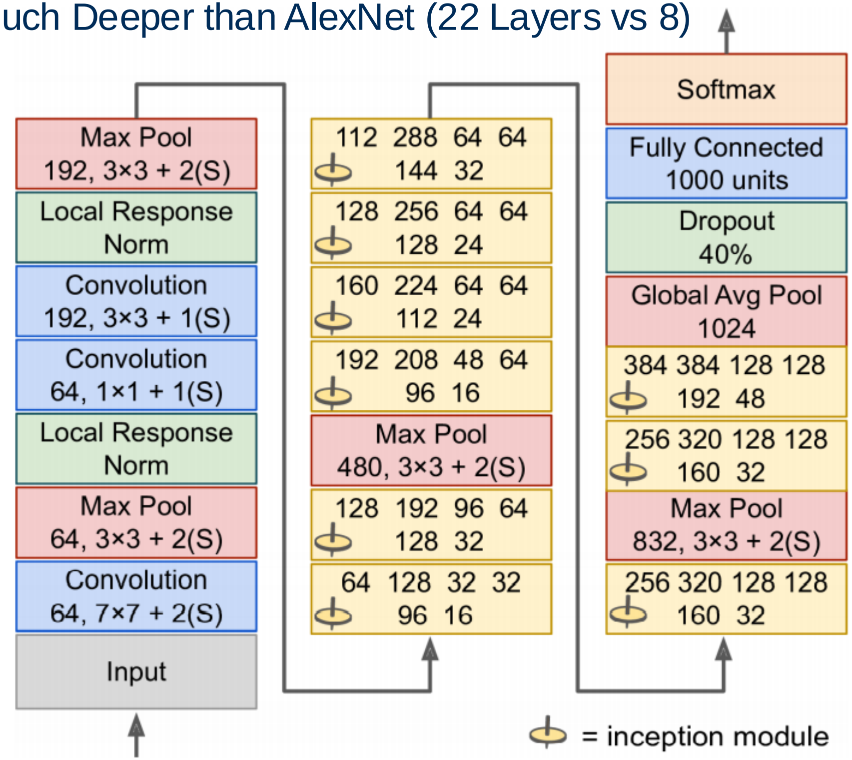
Test img

Training imgs with closest feature vector in last H-layer

Orig. Paper, “ImageNet Classification with Deep Convolutional Neural Networks,” A. Krizhevsky et al. (2012). https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

GoogLeNet (2014)

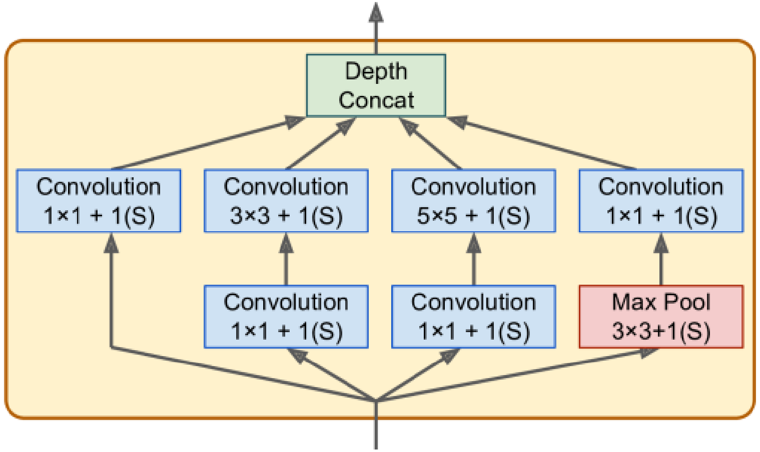
Much Deeper than AlexNet (22 Layers vs 8)



top-5 error rate ~ 7%

More efficient -> 6 M parameters

NEW: Inception Module



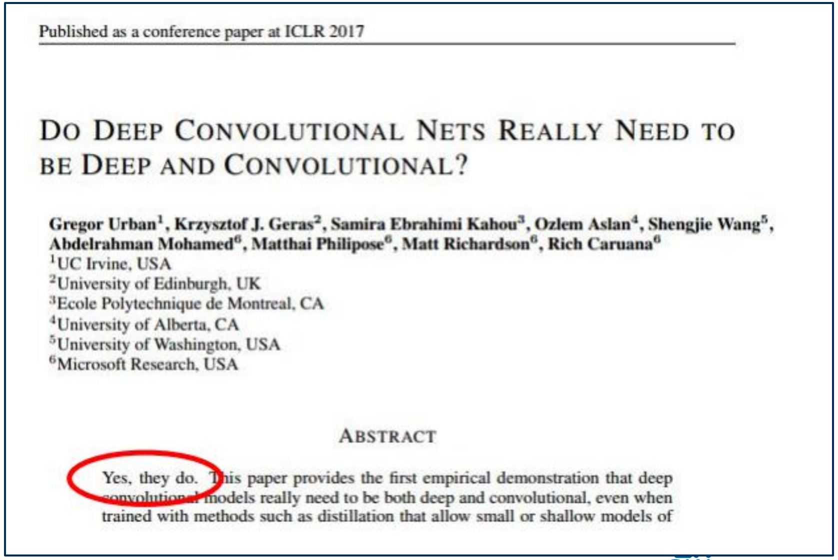
"Going Deeper with Convolutions," C. Szegedy et al. (2015).

https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf

ResNet (2015)

- ResidualNet
- Top5 Error rate ~ 3.6%
- Many Version (ResNet34, ResNet50…, ResNet152)
- Extremely Deep: 152 Layers
- “Deep Residual Learning for Image Recognition,” K. He (2015).

Trend: Deeper Networks, possibly fewer params


Published as a conference paper at ICLR 2017

DO DEEP CONVOLUTIONAL NETS REALLY NEED TO BE DEEP AND CONVOLUTIONAL?

Gregor Urban[1], Krzysztof J. Geras[2], Samira Ebrahimi Kahou[3], Ozlem Aslan[4], Shengjie Wang[5], Abdelrahman Mohamed[6], Matthai Philipose[6], Matt Richardson[6], Rich Caruana[6]
[1]UC Irvine, USA
[2]University of Edinburgh, UK
[3]Ecole Polytechnique de Montreal, CA
[4]University of Alberta, CA
[5]University of Washington, USA
[6]Microsoft Research, USA

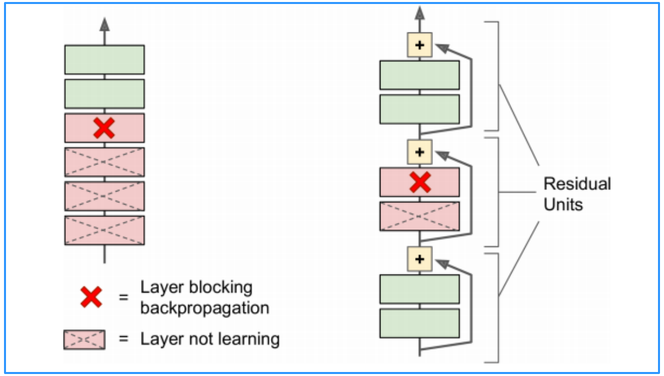
ABSTRACT

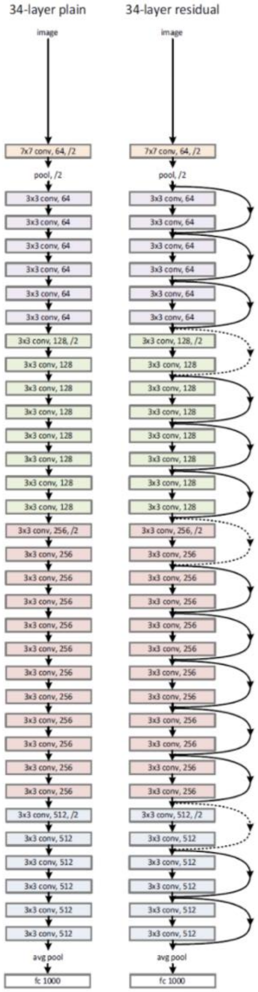
Yes, they do. This paper provides the first empirical demonstration that deep convolutional models really need to be both deep and convolutional, even when trained with methods such as distillation that allow small or shallow models of

ResNet (2015)

- ResidualNet
- Top5 Error rate ~ 3.6%
- by Kaiming He et al.
- Many Version, Extremely Deep: 152 Layers

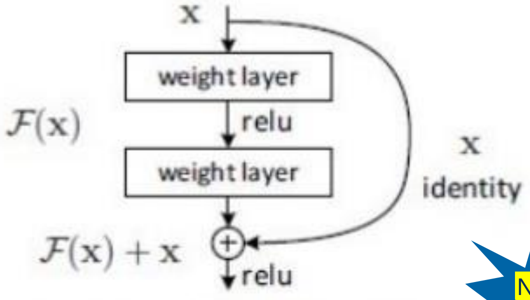
Many skip connections allow learning to advance faster through the network





Main novelty: Skip Connections

(aka) Residual Block



No extra params

- Initial weights ~ 0
- The output ~ 0
- BUT the skip connection add the input!

Output ~ Input

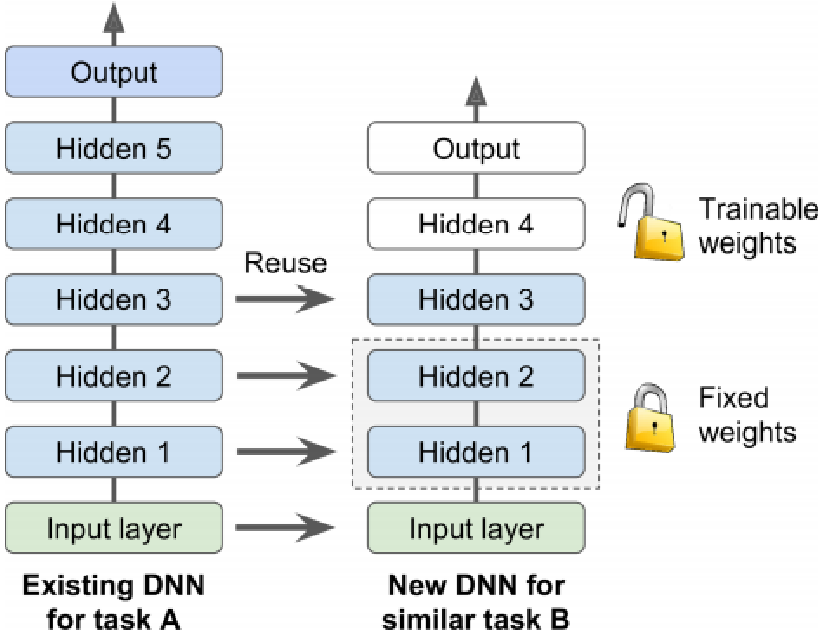
~ giving a Hint -> Speed up training!!

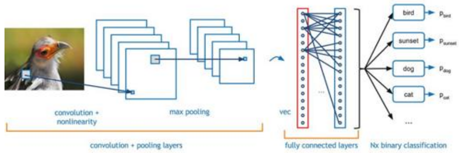
Img: Credits: https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8

Reusing Layers and Transfer Learning

Transfer Learning

Idea: Reusing already pretrained layers in similar tasks





- **Use Pretrained Models**
Simply use an already trained network on a similar task (but on way more data!)
- **ConvNet as feature extractors**
Keep only the conv layers of the network and add a new classifier
- **Fine-tuning of ConvNet**
Keep the conv layers, retrain the classifier but also fine tune the weights of the conv layers (at least top layers)

Transfer Learning: To fine-tune or not to fine-tune?

REMEMBER: ConvNet features are more generic in early layers and more original-dataset-specific in later layers!

Should I fine tune the pretrained conv layers? It depends on:

Size of the new dataset

Similarity to the original dataset

	Similar	Not Similar
Small	No Fine tune (overfit + no need due to similarity)	No Fine tune (overfit) Better include only earlier layers (less dataset specific)
Large	Maybe no need but fine tune is ok	Good idea to fine tune

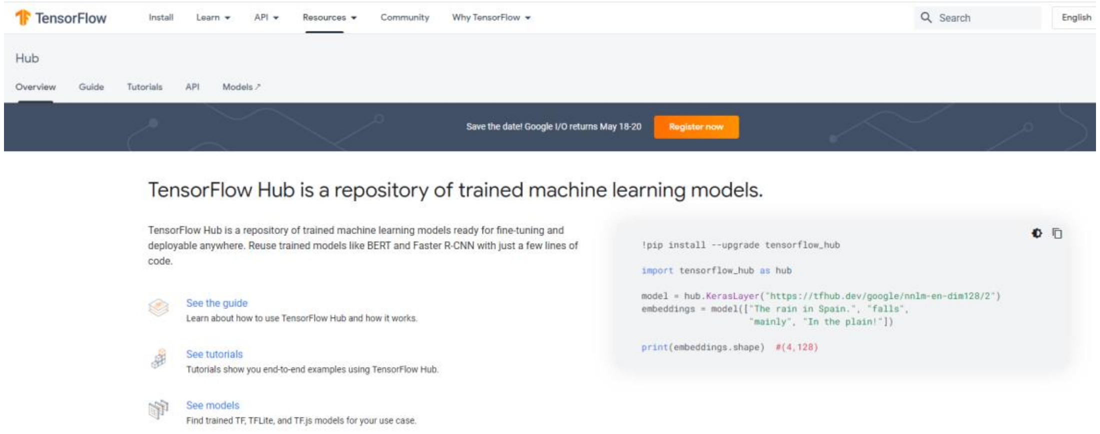
Also do not fine tune straight from scratch! first Transfer Learning and then fine tune

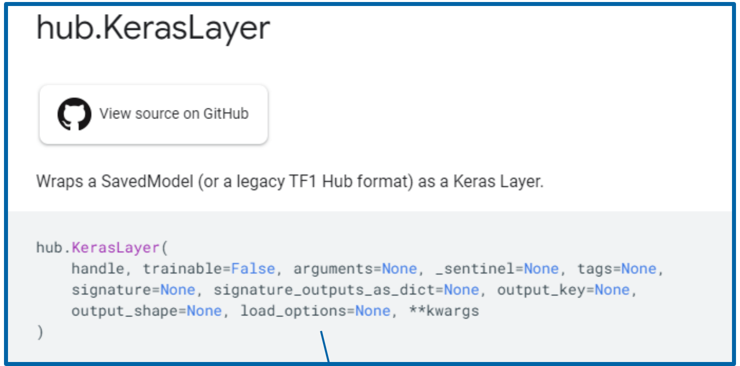
Advice about Learning rates: Use SMALL learning rate for ConvNet weights that are being fine-tuned! We don’t want to break them, they should be quite good already…

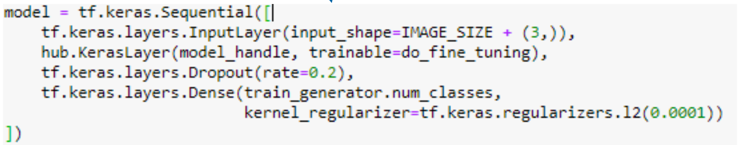
Source: https://cs231n.github.io/transfer-learning/

TensorflowHUB

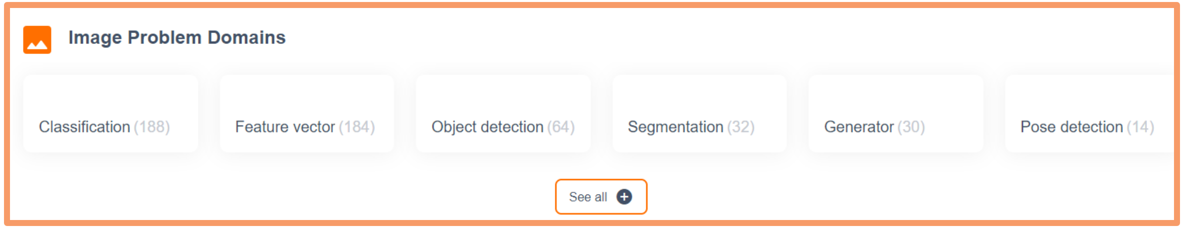
https://www.tensorflow.org/hub







List of models:
https://tfhub.dev/



zh aw School of Management and Law

Module: tf.keras.applications

https://www.tensorflow.org/api_docs/python/tf/keras/applications

Modules

`densenet` module: DenseNet models for Keras.

`efficientnet` module: EfficientNet models for Keras.

`imagenet_utils` module: Utilities for ImageNet data preprocessing & prediction decoding.

`inception_resnet_v2` module: Inception-ResNet V2 model for Keras.

`inception_v3` module: Inception V3 model for Keras.

`mobilenet` module: MobileNet v1 models for Keras.

`mobilenet_v2` module: MobileNet v2 models for Keras.

`mobilenet_v3` module: MobileNet v3 models for Keras.

`nasnet` module: NASNet-A models for Keras.

`resnet` module: ResNet models for Keras.

`resnet50` module: Public API for tf.keras.applications.resnet50 namespace.

`resnet_v2` module: ResNet v2 models for Keras.

`vgg16` module: VGG16 model for Keras.

`vgg19` module: VGG19 model for Keras.

```
base_model = keras.applications.xception.Xception(weights="imagenet",
                                                  include_top=False)
avg = keras.layers.GlobalAveragePooling2D()(base_model.output)
output = keras.layers.Dense(n_classes, activation="softmax")(avg)
model = keras.models.Model(inputs=base_model.input, outputs=output)
```

```
for layer in base_model.layers:
    layer.trainable = False
```

```
optimizer = keras.optimizers.SGD(lr=0.2, momentum=0.9, decay=0.01)
model.compile(loss="sparse_categorical_crossentropy", optimizer=optimizer,
              metrics=["accuracy"])
history = model.fit(train_set,
                    steps_per_epoch=int(0.75 * dataset_size / batch_size),
                    validation_data=valid_set,
                    validation_steps=int(0.15 * dataset_size / batch_size),
                    epochs=5)
```

REMEMBER: you need to preprocess the input accordingly! Check the Keras Guidebook!

Visual Data preparation

Data Augmentation

Idea: artificially increase the training set size/diversity by **generating variants** of training examples

- Random Cropping
- Mirroring/Flipping (viewpoint invariance)
- Color shifting
- Resizing/scaling
- Rotation
- Variation in contrast (illumination invariance)

<u>Realistic generation!</u>
A human should not be able to tell it was augmented

(Effect: ~Regularization)

Data Augmentation

How to do it in Keras/TF?

1. Keras Image Data Generator
tf.keras.preprocessing.image.ImageDataGenerator (Data Augmentation Class)
(https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator)

```
tf.keras.preprocessing.image.ImageDataGenerator(
    featurewise_center=False, samplewise_center=False,
    featurewise_std_normalization=False, samplewise_std_normalization=False,
    zca_whitening=False, zca_epsilon=1e-06, rotation_range=0, width_shift_range=0.0,
    height_shift_range=0.0, brightness_range=None, shear_range=0.0, zoom_range=0.0,
    channel_shift_range=0.0, fill_mode='nearest', cval=0.0,
    horizontal_flip=False, vertical_flip=False, rescale=None,
    preprocessing_function=None, data_format=None, validation_split=0.0, dtype=None
)
```

2. **Keras Preprocessing Layers**
tf.keras.layers.experimental.preprocessing
(https://www.tensorflow.org/api_docs/python/tf/keras/layers/)

- RandomContrast
- RandomCrop
- RandomFlip
- RandomHeight
- RandomRotation
- RandomTranslation
- RandomWidth
- RandomZoom
- Rescaling
- Resizing

3. **Keras Preprocessing Utils**
tf.keras.preprocessing.image
(https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image)

- apply_affine_transform
- apply_brightness_shift
- apply_channel_shift
- array_to_img
- img_to_array
- load_img
- random_brightness
- random_channel_shift
- random_rotation
- random_shear
- random_shift
- random_zoom
- save_img
- smart_resize

4. **TF Image Utils**
tf.image
(https://www.tensorflow.org/api_docs/python/tf/image)

https://www.tensorflow.org/tutorials/images/data_augmentation

Life after Image Classification…

Computer Vision Tasks

Image Generation

Image Colorisation/ Reconstruction /super Resolution

Image Captioning / Analysis

Pose Estimation

Visual Question Answering

Classification

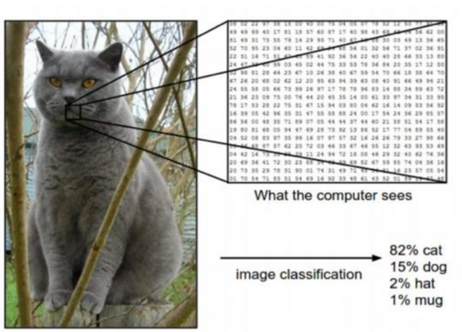
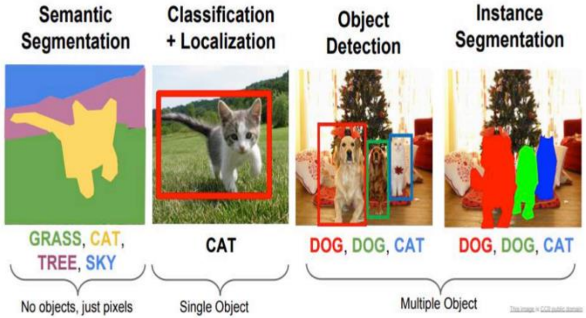


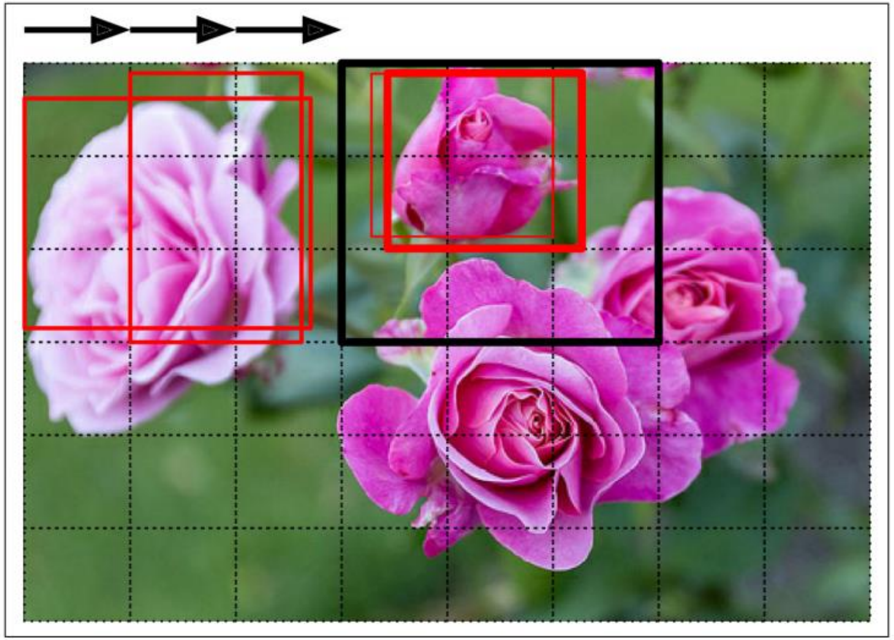
Image Segmentation

Object Detection



Object Detection

= The task of classifying and localizing multiple objects in an image



Several Obj. Detection Models:

- YOLO (“You only look once”)
- SSD (Single Shot Detectors)
- Faster-RCNN

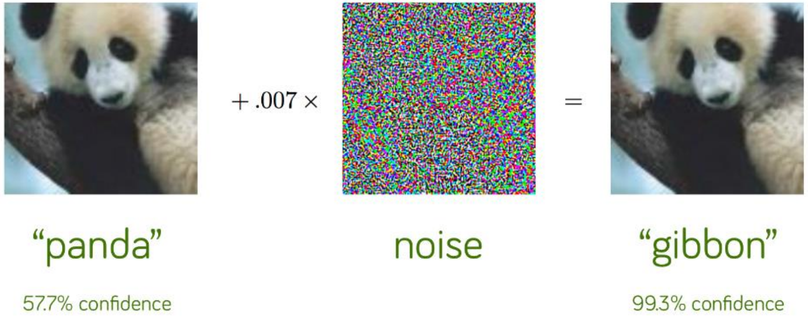
– “You Only Look Once: Unified, Real-Time Object Detection,” J. Redmon, S. Divvala, R. Girshick, A. Farhadi (2015).
– “SSD: Single Shot MultiBox Detector,” Wei Liu et al. (2015). 30
– “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” Shaoqing Ren et al. (2015).

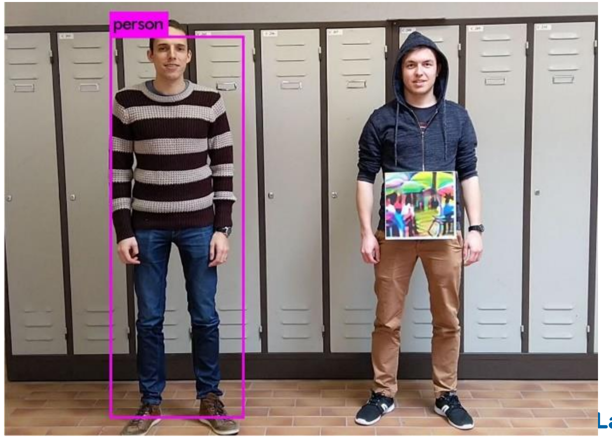
Limitations & Challenges

Is it a Person?

Limitations & Challenges

Adversial Patches





The role of the environment

(A) **Cow: 0.99**, Pasture: 0.99, Grass: 0.99, No Person: 0.98, Mammal: 0.98

(B) No Person: 0.99, Water: 0.98, Beach: 0.97, Outdoors: 0.97, Seashore: 0.97

(C) No Person: 0.97, **Mammal: 0.96**, Water: 0.94, Beach: 0.94, Two: 0.94

https://arxiv.org/pdf/1807.04975.pdf

Vision Transformers & Self-Supervision

Transformers and “Attention is all you need” (2017)

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

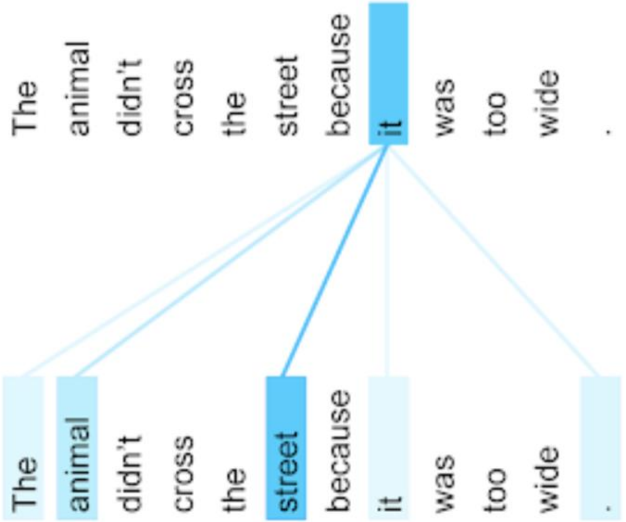
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

- Long memory
- Long temporal/spatial dependencies
- Good for parallelization

↓

All LLM we know



https://www.tensorflow.org/text/tutorials/transformer

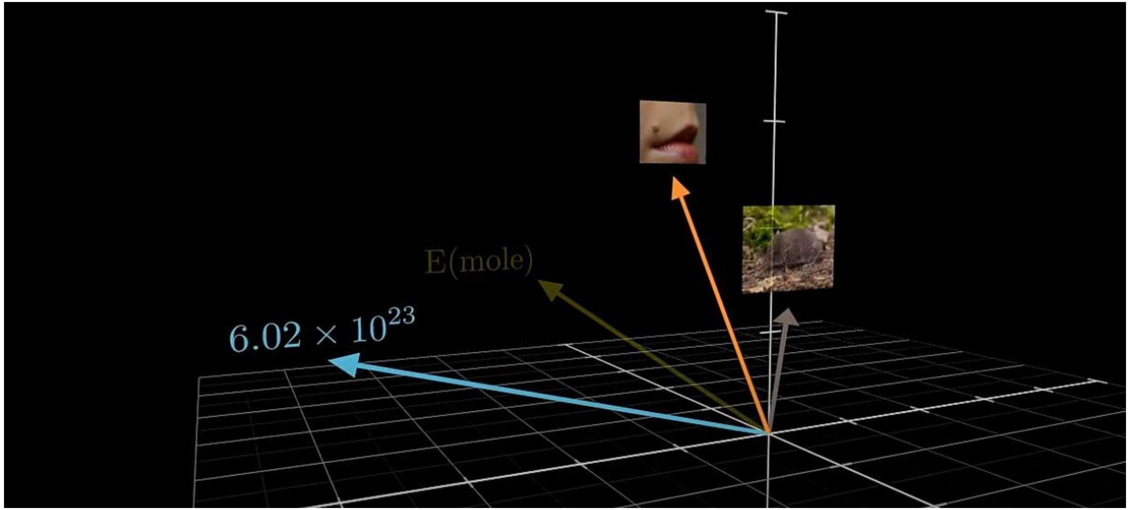
Attention

Example:

"American Shrew **mole**"

"One **mole** of carbon dioxide"

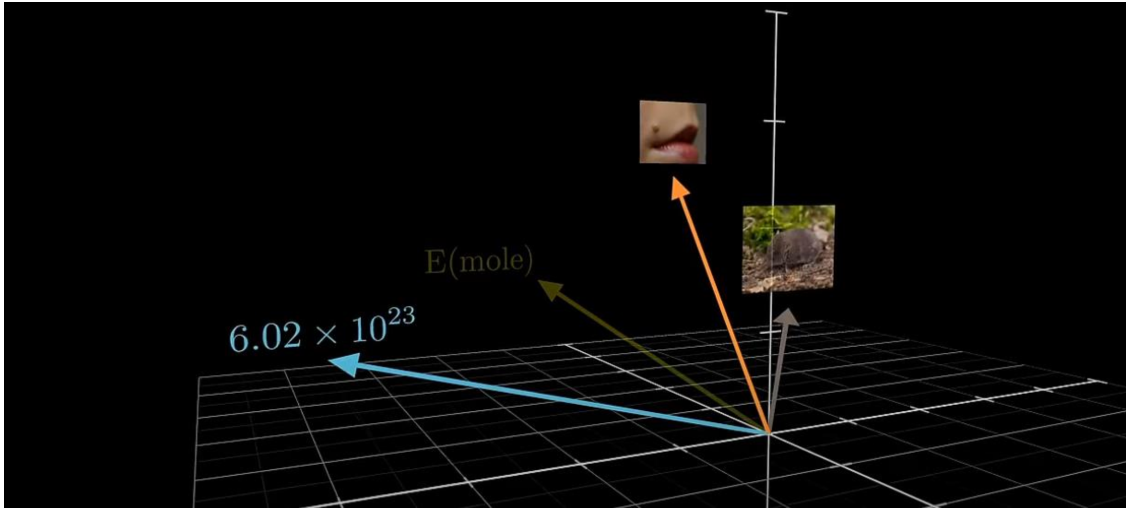
"Take a biopsy of the **mole**"



Attention

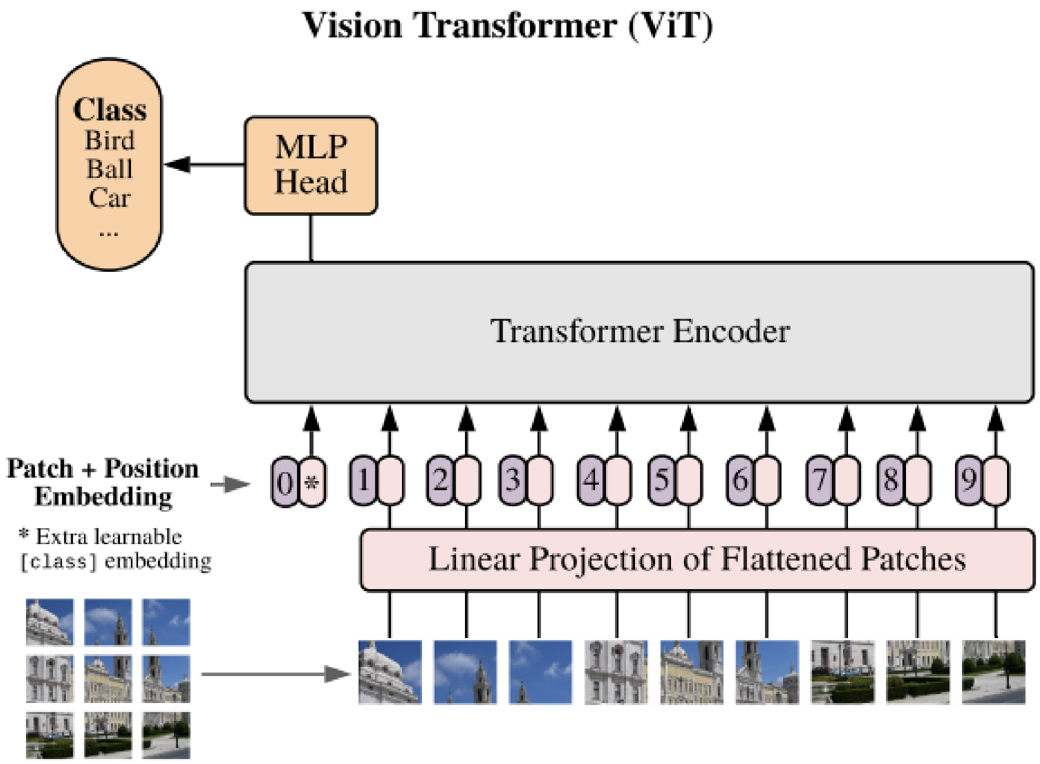
Reference (s. Exercise)
https://www.3blue1brown.com/lessons/attention

…self-attention (vs cross-attention) & multi-head



Interactive Playground: https://poloclub.github.io/transformer-explainer/

Vision Transformers – ViT (2020)

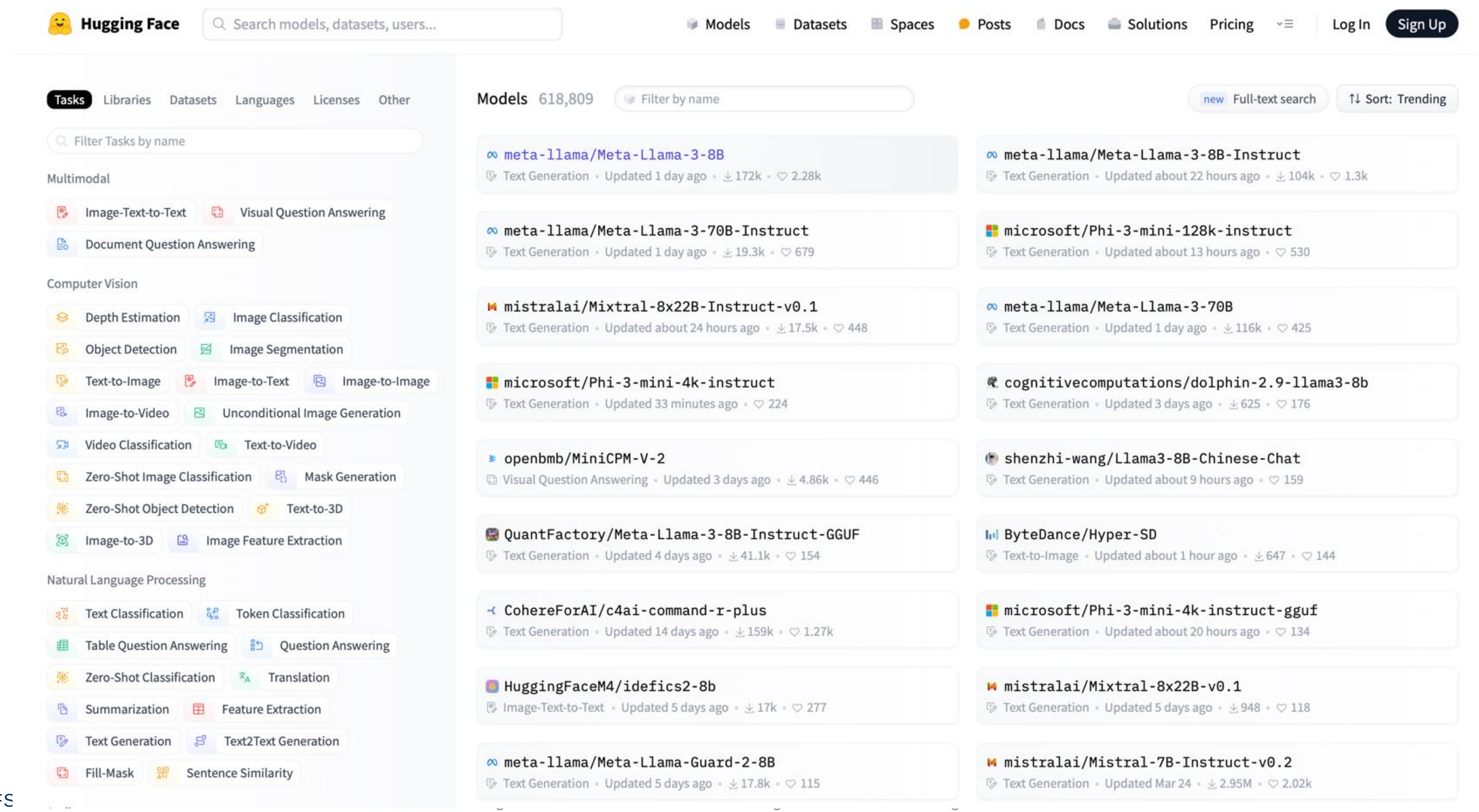


https://arxiv.org/pdf/2010.11929v2.pdf

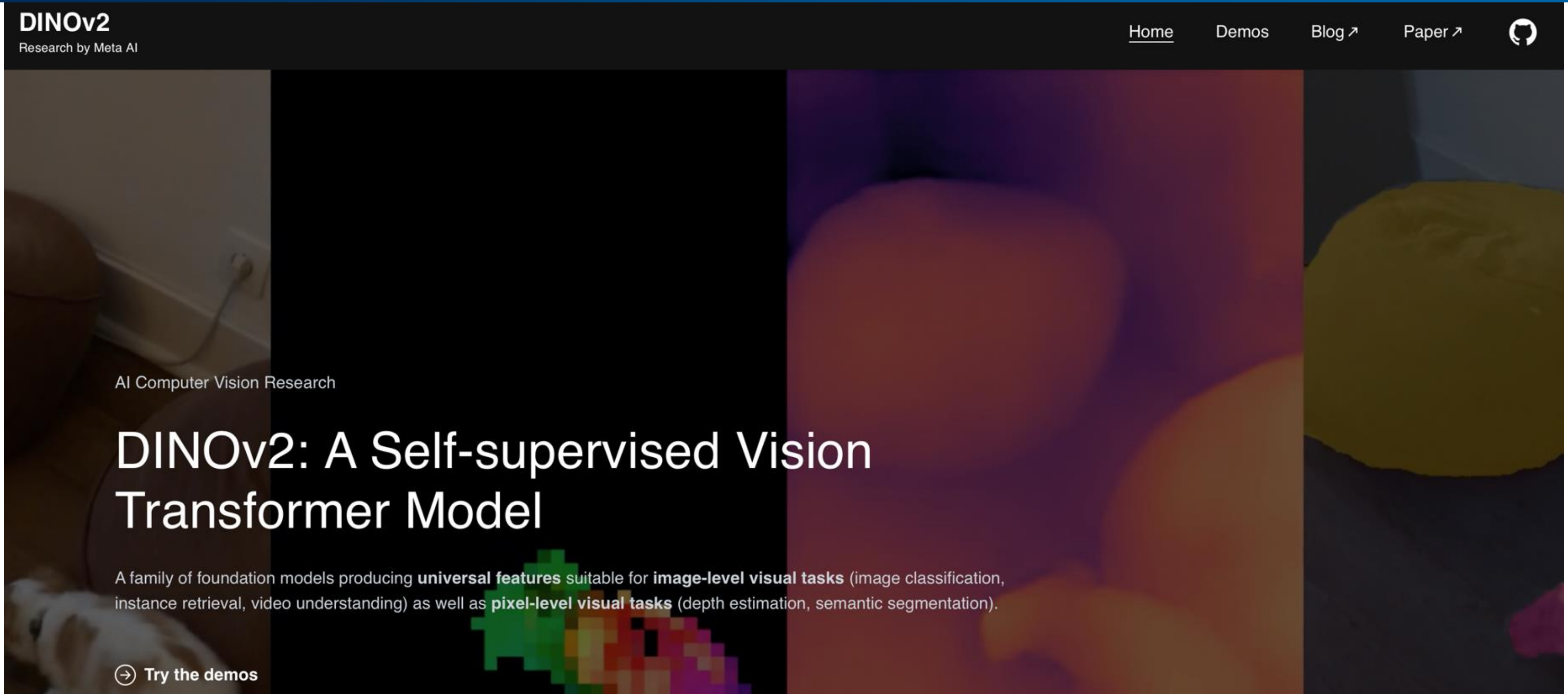
https://github.com/google-research/vision_

Hugging Face

https://huggingface.co

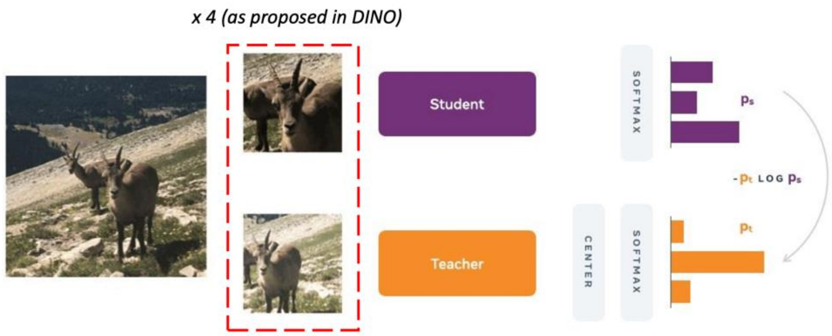


DINOv2 (2021/2023)



https://dinov2.metademolab.com

DINO: Emerging Properties in Self-Supervised Vision Transformers



Orig Paper: https://arxiv.org/pdf/2104.14294.pdf

Source: https://lmb.informatik.uni-freiburg.de/lectures/seminar_brox/seminar_ws2122/dino_presentation.pdf

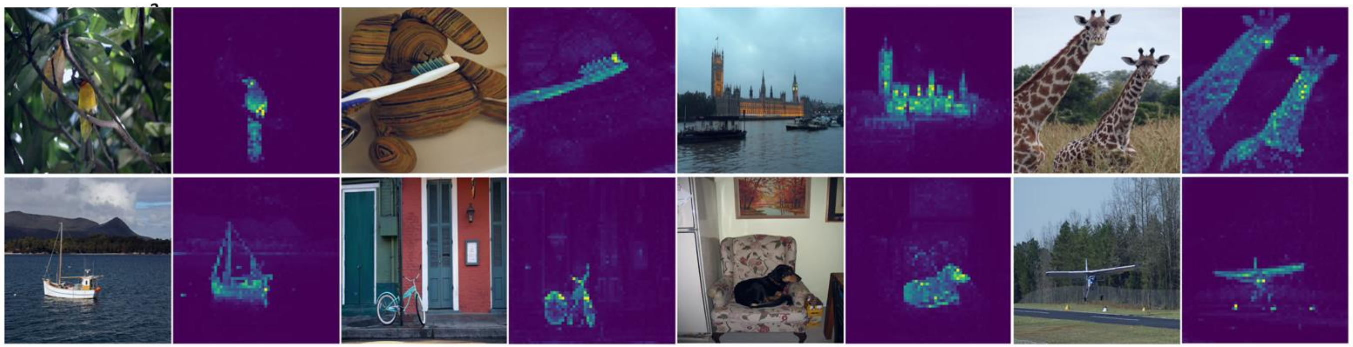


Figure 1: **Self-attention from a Vision Transformer with 8×8 patches trained with no supervision.** We look at the self-attention of the [CLS] token on the heads of the last layer. This token is not attached to any label nor supervision. These maps show that the model automatically learns class-specific features leading to unsupervised object segmentations.

School of Management and Law

Useful Reads:

A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, Tensorflow*, O'Reilly
Chapter 14 (& 15)

- Additional Tensorflow Tutorials:

- https://www.tensorflow.org/hub/tutorials/image_classification
- https://www.tensorflow.org/tutorials/images/transfer_learning_with_hub
- https://www.tensorflow.org/hub/tutorials/tf2_image_retraining
- https://www.tensorflow.org/tutorials/images/data_augmentation
- https://huggingface.co/docs/transformers/model_doc/vit
- https://huggingface.co/blog/encoder-decoder
- https://huggingface.co/learn/computer-vision-course/en/unit3/vision-transformers/vi transformers-for-image-classification#multi-class-image-classification
- https://colab.research.google.com/github/huggingface/notebooks/blob/main/examp e_classification.ipynb

Thank you.