

Lecture 6: Computer Vision (part 3)

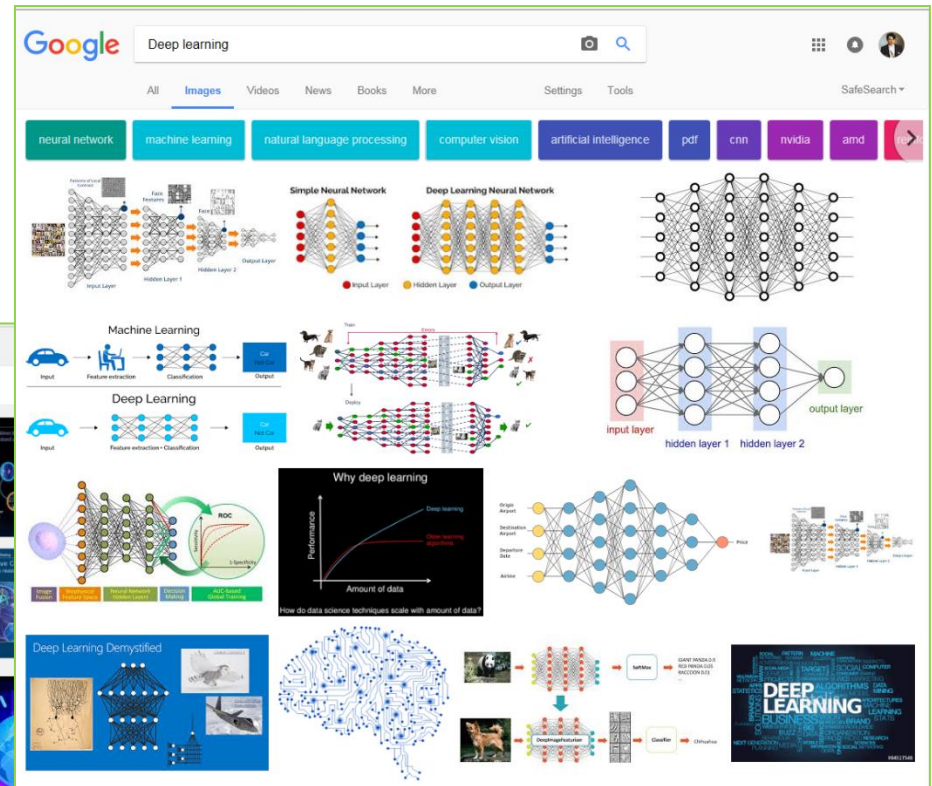
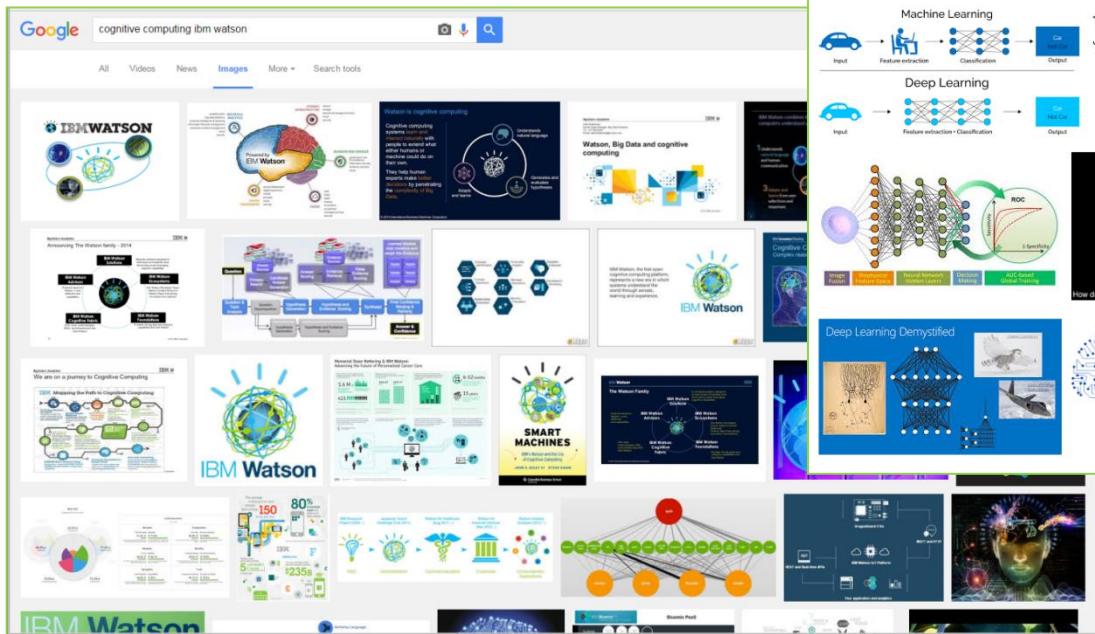
Image Segmentation and Neural Style Transfer

TIES4911 Deep-Learning for Cognitive Computing for Developers
Spring 2024

by:
Dr. Oleksiy Khriyenko
IT Faculty
University of Jyväskylä

Acknowledgement

I am grateful to all the creators/owners of the images that I found from Google and have used in this presentation.



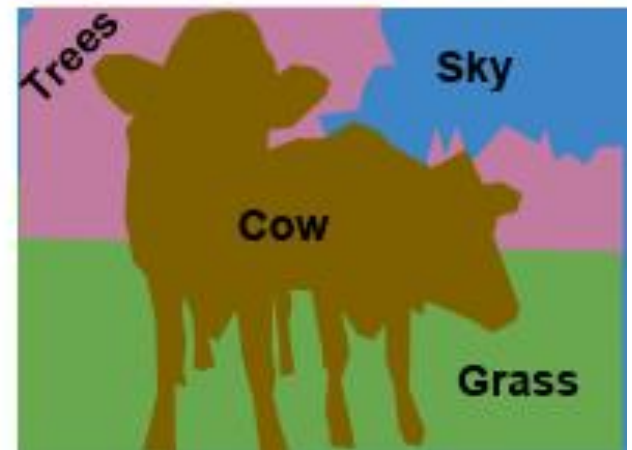
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

Semantic Segmentation



*The goal is to associate each pixel with a category label.
At this point the model does not differentiate instances...*

Relevant links:

<https://arxiv.org/pdf/2001.05566.pdf>

29/02/2024

Semantic Segmentation

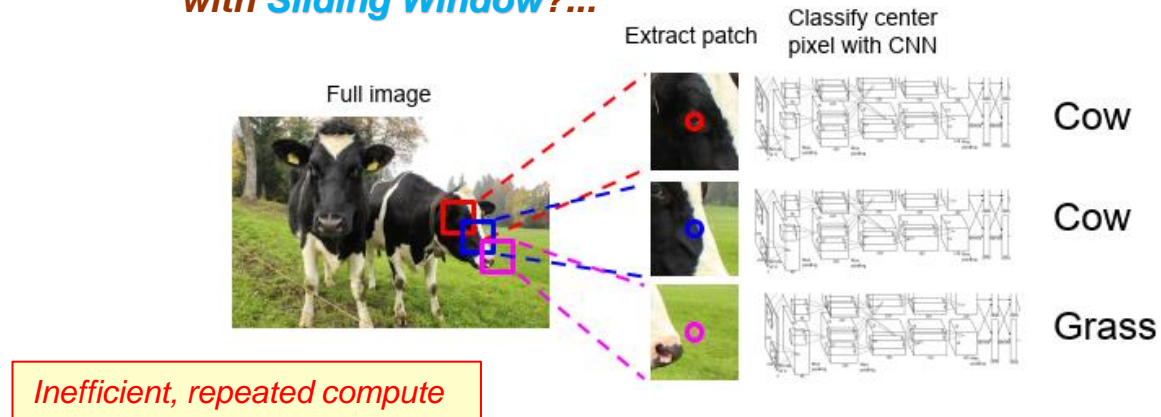
Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

Can we approach Semantic Segmentation with Sliding Window?...



Relevant links:

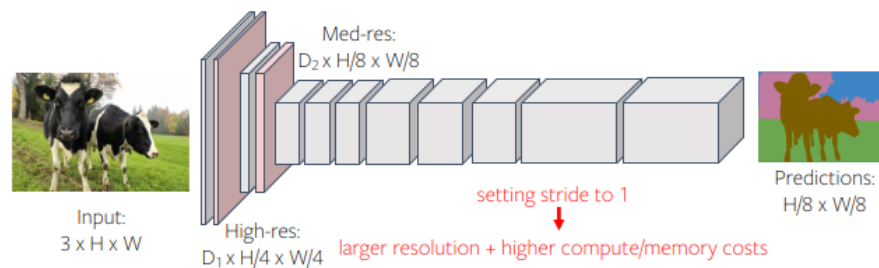
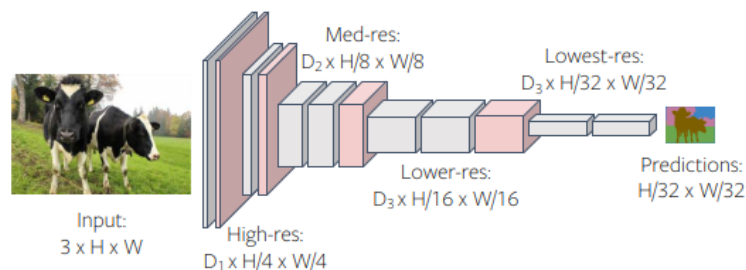
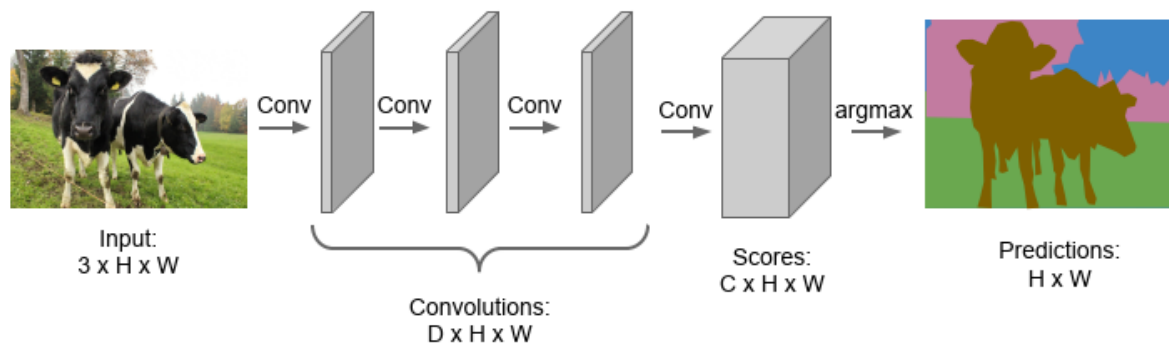
<https://ieeexplore.ieee.org/document/6338939>

<http://proceedings.mlr.press/v32/pinheiro14.pdf>

Semantic Segmentation

Can we approach Semantic Segmentation with Fully Convolutional NN ?...

Design a network as a bunch of convolutional layers to make predictions for pixels all at once! Using Per-Pixel cross-entropy loss function...



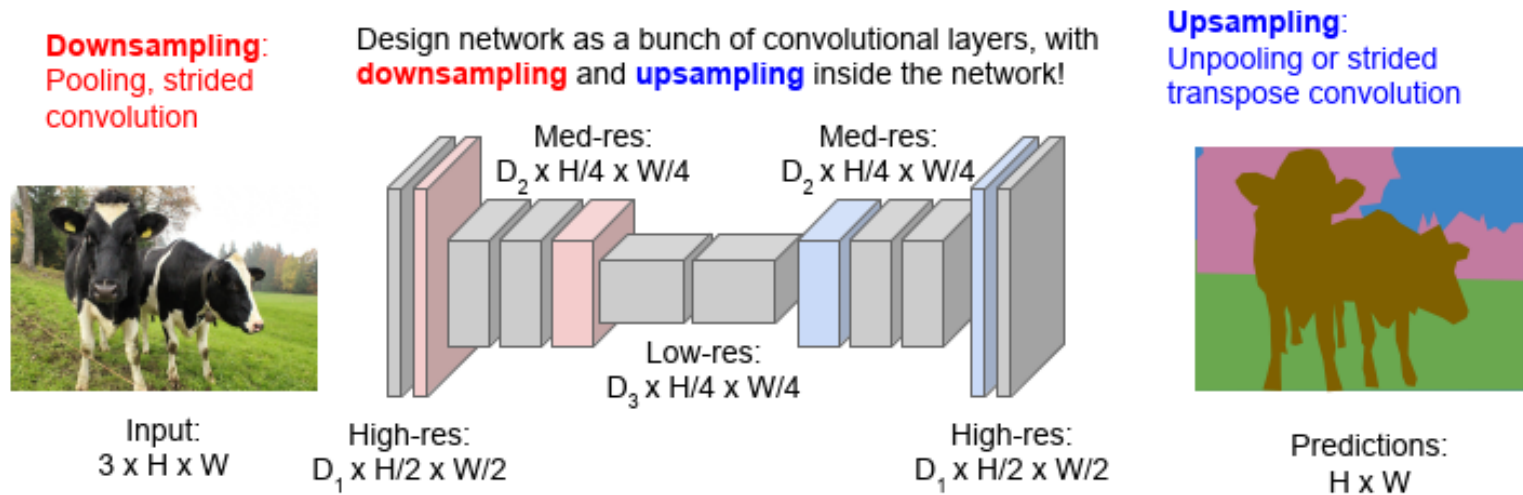
Relevant links:

<https://ieeexplore.ieee.org/document/6338939>
<http://proceedings.mlr.press/v32/pinheiro14.pdf>

29/02/2024

Semantic Segmentation

Encoder-Decoder Models for General Segmentation (e.g. SegNet, HRNet)



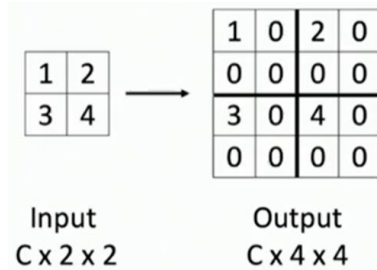
Relevant links:

<https://arxiv.org/abs/1411.4038> , <https://arxiv.org/abs/1505.04366>
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

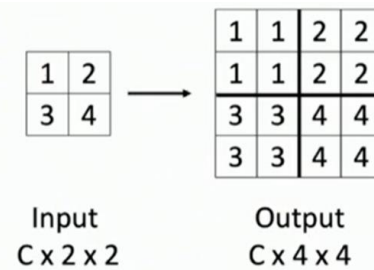
29/02/2024

Semantic Segmentation

Bed of Nails Unpooling

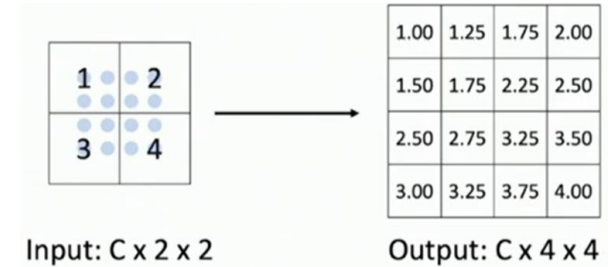


Nearest Neighbor Unpooling

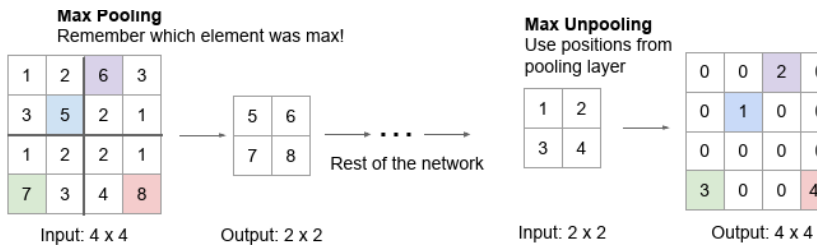


Bilinear/Bicubic Interpolation

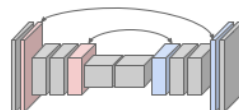
(using two/three closest neighbors to construct corresponding approximations)



Max Unpooling

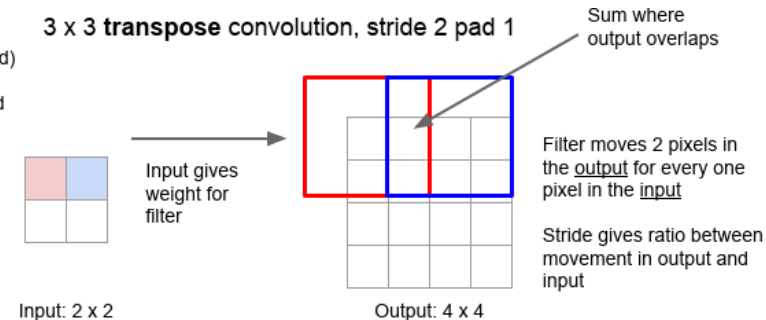


Corresponding pairs of downsampling and upsampling layers



Other names:
-Deconvolution (bad)
-Upconvolution
-Fractionally strided convolution
-Backward strided convolution

Learnable upsampling:



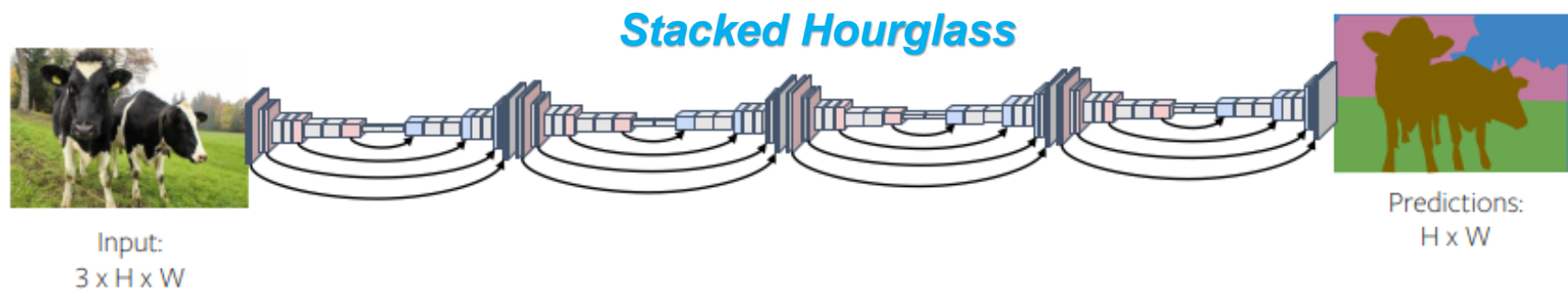
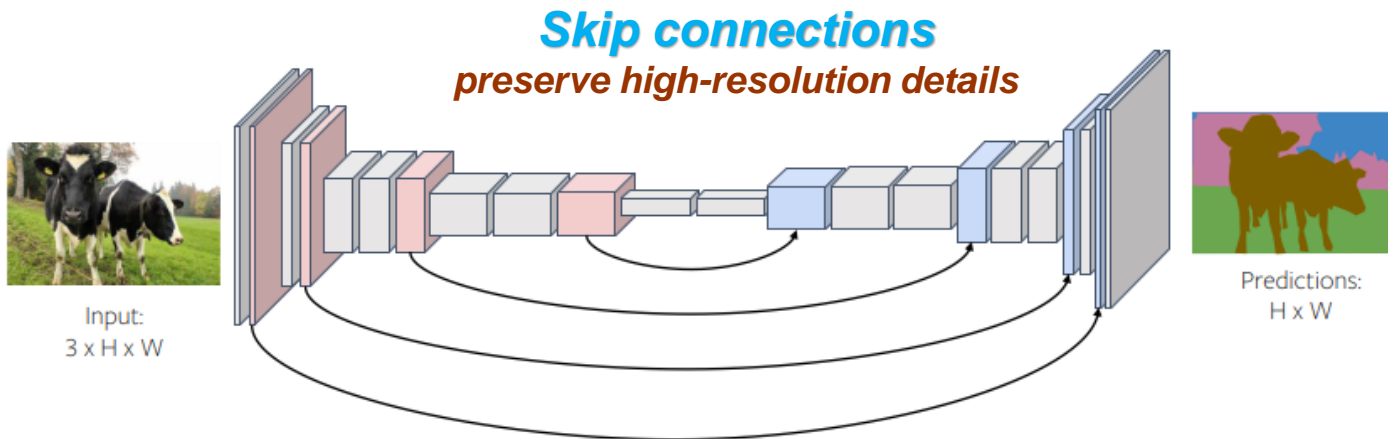
Relevant links:

<https://arxiv.org/abs/1411.4038> , <https://arxiv.org/abs/1505.04366>
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

29/02/2024

Semantic Segmentation

Encoder-Decoder Models for General Segmentation (e.g. SegNet, HRNet)



Relevant links:

<https://arxiv.org/abs/1505.04597>

<https://arxiv.org/abs/1603.06937>

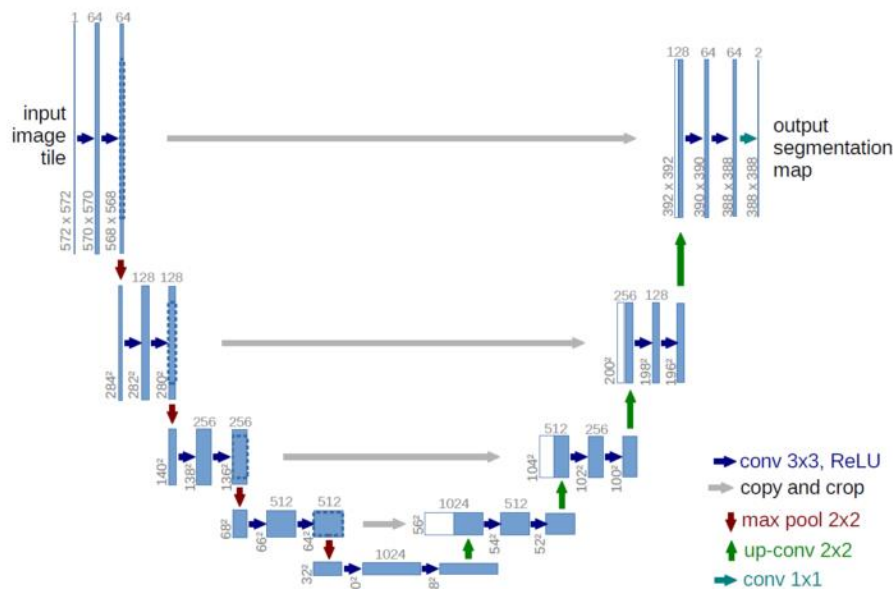
29/02/2024

TIES4911 – Lecture 6

Semantic Segmentation

Encoder-Decoder Models for General Segmentation (e.g. U-Net, V-Net)

U-Net



V-Net

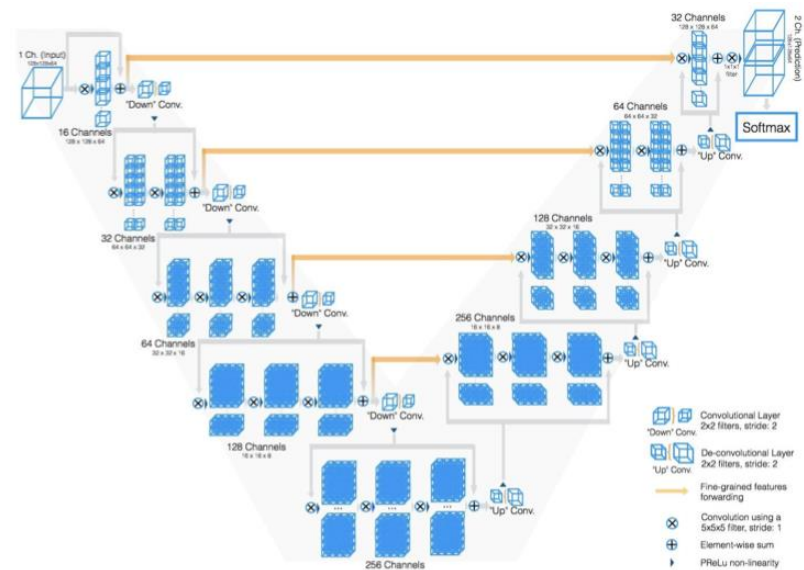


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Relevant links:

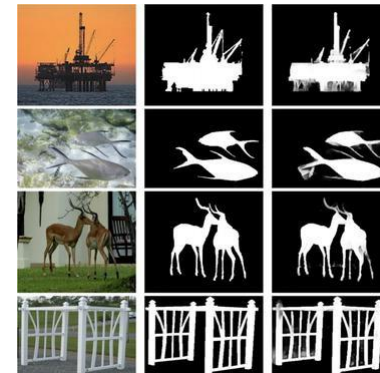
<https://arxiv.org/abs/1505.04597>

<https://arxiv.org/abs/1606.04797>

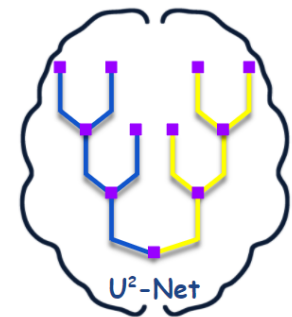
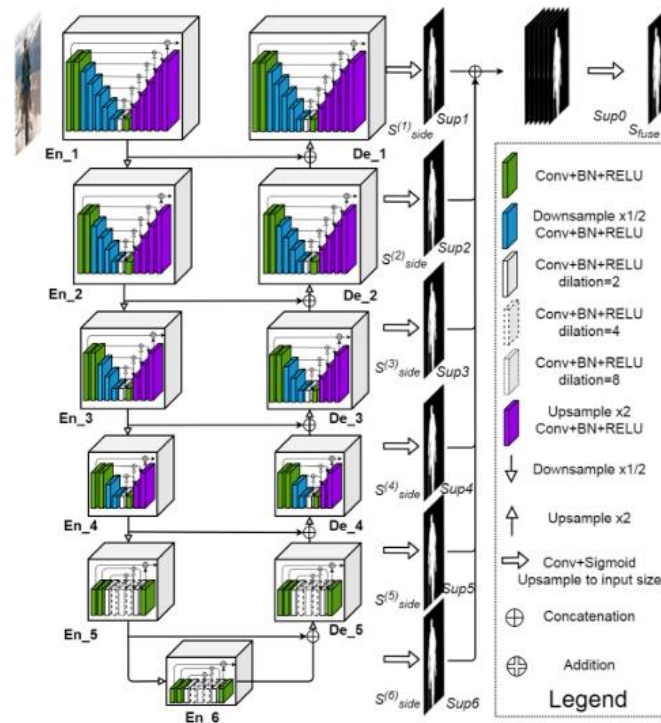
Semantic Segmentation

Nested U-Structure for Salient Object Detection (U²-Net, 2020)

U²-Net was designed for the purpose of **Saliency Object Detection (SOD)** – the task based on a visual attention mechanism, in which algorithms aim to explore objects or regions more attentive than the surrounding areas on the scene or images (it is basically detecting the most important or the main object in a given image).



U²-Net (U-squared Net) is a two-level nested U-structure architecture that allows the network to go deeper, attain high resolution, without significantly increasing the memory and computation cost. This is achieved by a nested U-structure: on the bottom level, with a novel **ReSidual U-block (RSU) module**, which is able to extract intra-stage multi-scale features without degrading the feature map resolution; on the top level, there is a U-Net like structure, in which each stage is filled by a RSU block.



Relevant links:

<https://arxiv.org/abs/2005.09007>

<https://github.com/xuebinjin/U-2-Net>

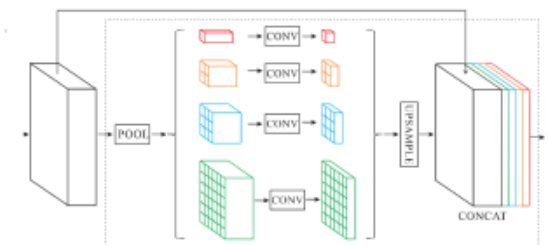
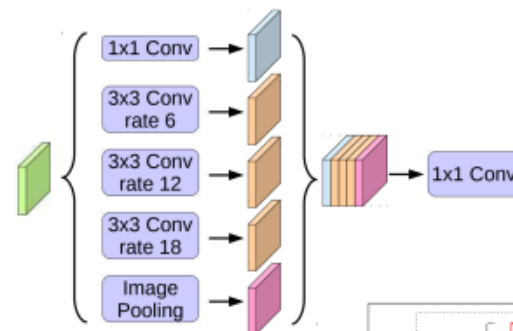
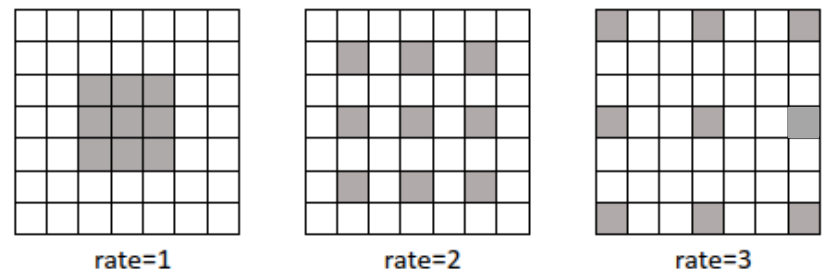
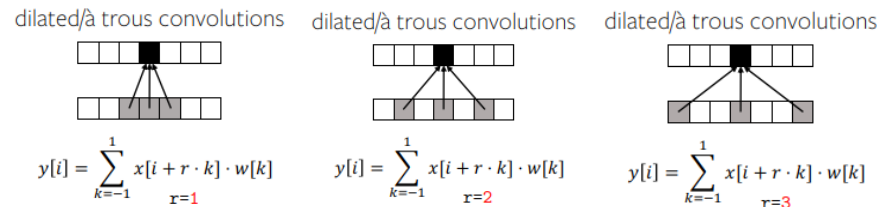
Figure 5. Illustration of our proposed U²-Net architecture. The main architecture is a U-Net like Encoder-Decoder, where each stage consists of our newly proposed residual U-block (RSU). For example, **En₁** is based on our RSU block shown in Fig. 2(e). Detailed configuration of RSU block of each stage is given in the last two rows of Table 1.

Semantic Segmentation

Multi-Scale and Pyramid Network Based Models (e.g. FPN, PSPN)



Context...



Relevant links:

<https://arxiv.org/abs/1802.02611>

<https://arxiv.org/abs/1612.01105>

29/02/2024

Instance Segmentation

Instance Segmentation

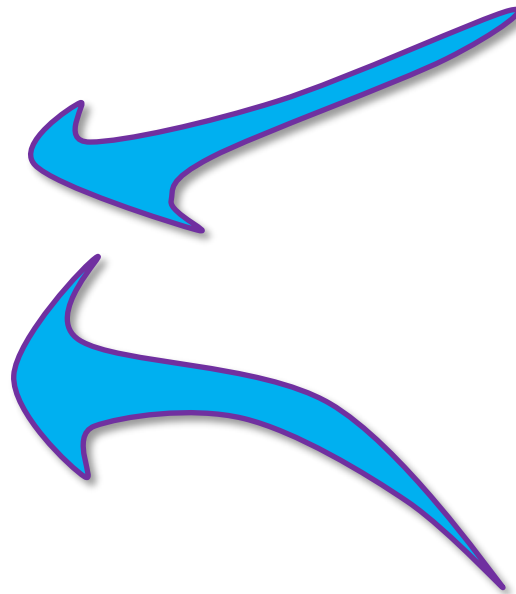
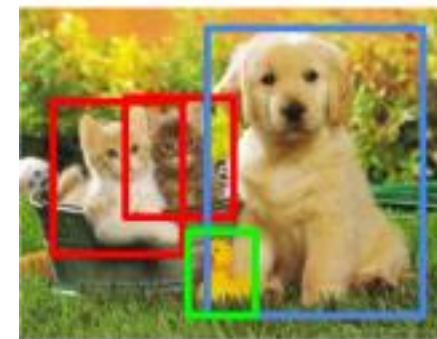


CAT, CAT, DOG, DUCK

Semantic Segmentation



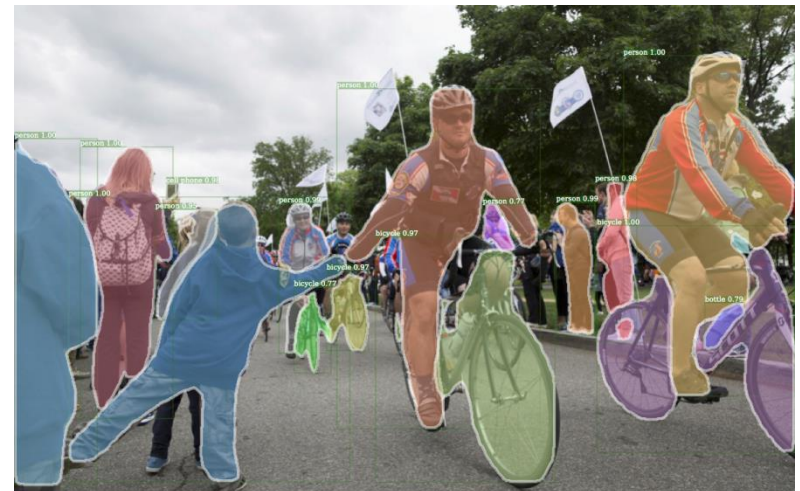
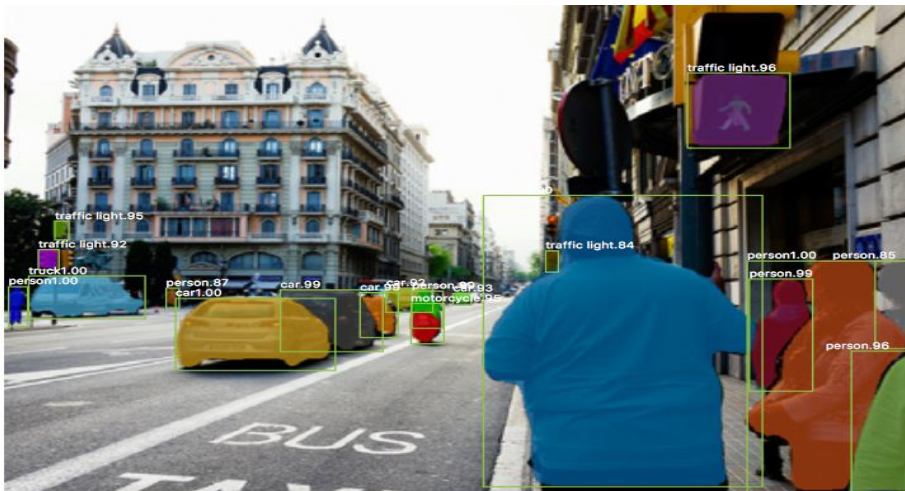
Object Detection



Mask R-CNN architecture

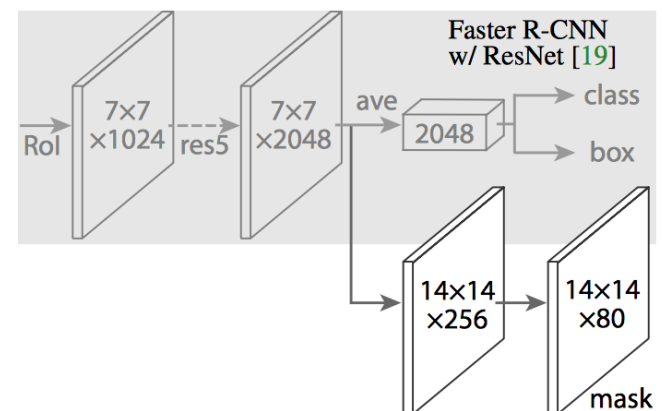
Mask R-CNN (team led by Kaiming He, a researcher at Facebook AI, 2017)

*Mask R-CNN extends Faster R-CNN for **Pixel Level Segmentation** that at a pixel level identifies what the different objects in a scene are ...*



In **Mask R-CNN**, a Fully Convolutional Network (FCN) is added on top of the CNN feature map of Faster R-CNN to generate a segmentation output - a binary mask (with 1s and 0s) that says whether or not a given pixel is part of an object. It is done in parallel to the classification and bounding box regression network of Faster R-CNN.

Mask R-CNN has one small adjustment by realigning *RoI/Pool* to be more accurate *RoIAlign*



Relevant links:

<https://arxiv.org/abs/1703.06870>

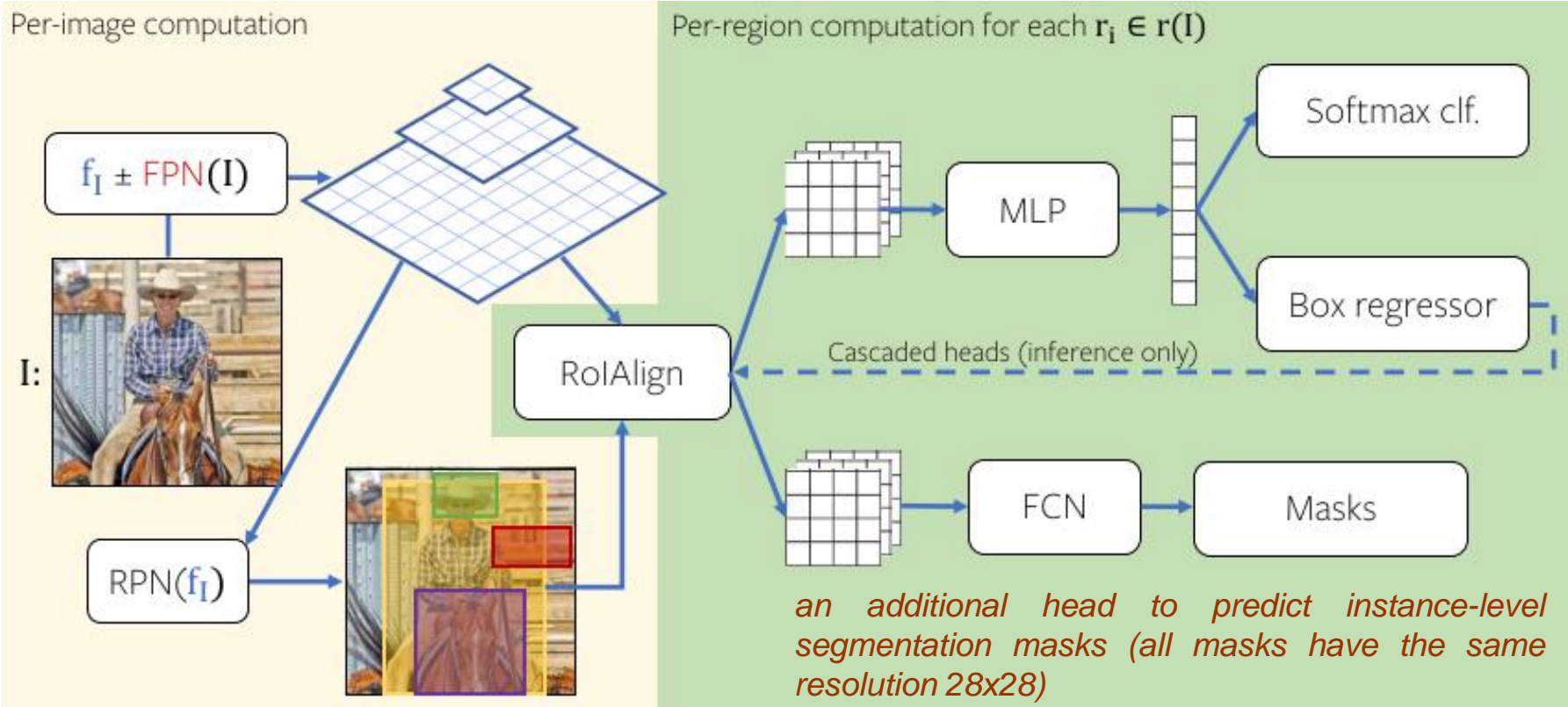
<https://github.com/CharlesShang/FastMaskRCNN>

https://github.com/matterport/Mask_RCNN

<https://github.com/facebookresearch/Detectron>

29/02/2024

Mask R-CNN architecture



Relevant links:

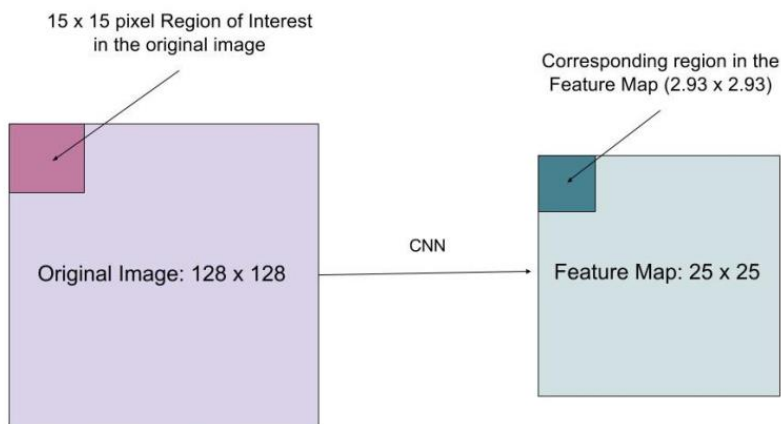
<https://www.youtube.com/watch?v=5bWG3ySf4fl>
<https://arxiv.org/abs/1703.06870>
<https://github.com/CharlesShang/FastMaskRCNN>
https://github.com/matterport/Mask_RCNN
<https://github.com/facebookresearch/Detectron>

29/02/2024

Mask R-CNN architecture

RoIAlign in Mask R-CNN (team led by Kaiming He, a researcher at Facebook AI, 2017)

The regions of the feature map selected by *RoIPool* in Faster R-CNN were slightly misaligned from the regions of the original image. Since pixel level segmentation requires more fine-grained alignment than bounding boxes, authors solved this problem by cleverly adjusting *RoIPool* to be more precisely aligned using a method known as *RoIAlign*.

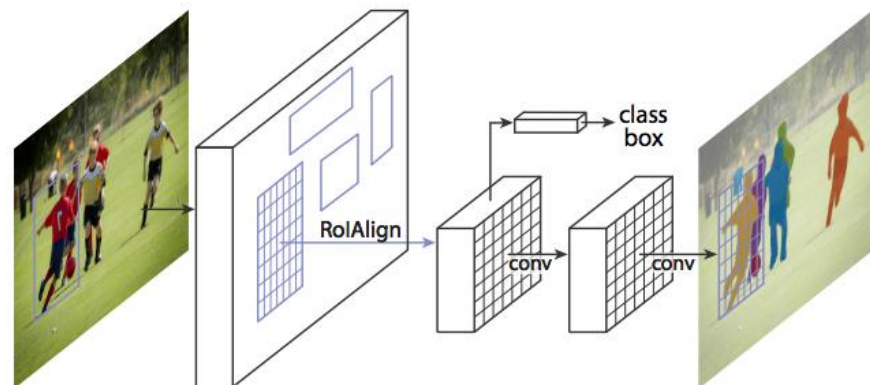


In *RoIPool*, we would round 2.93 down and select 2 pixels causing a slight misalignment. In *RoIAlign* avoids such rounding and uses *bilinear interpolation* to get a precise idea of what would be at pixel 2.93.

Relevant links:

<https://arxiv.org/abs/1703.06870>

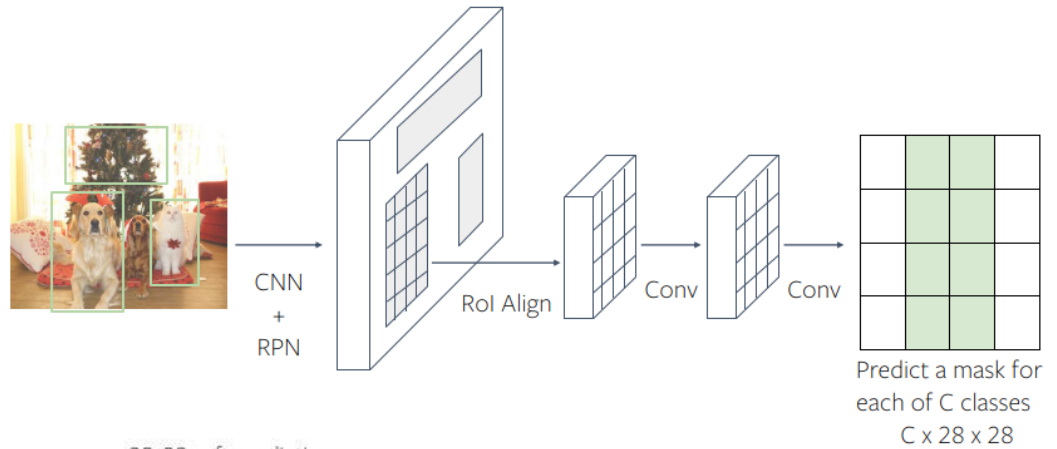
Bilinear interpolation: https://en.wikipedia.org/wiki/Bilinear_interpolation



Mask R-CNN combines generated masks with the classifications and bounding boxes from Faster R-CNN to generate precise segmentations.



Mask R-CNN architecture



Mask Prediction



28x28 soft prediction



Resized soft prediction



Final mask



Relevant links:

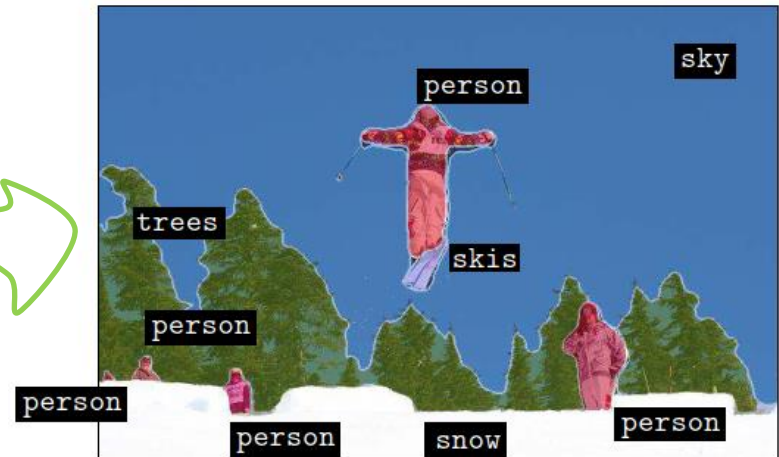
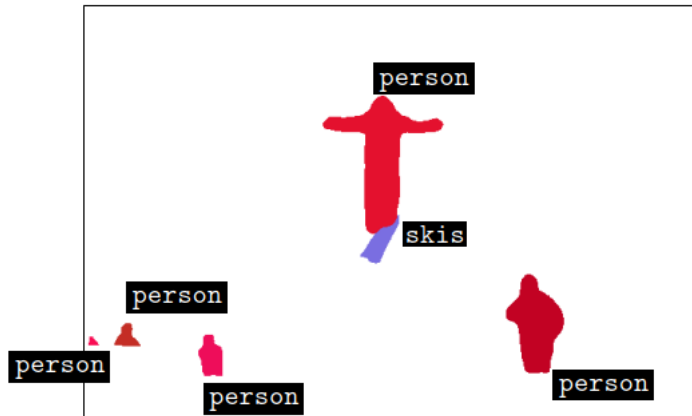
<https://www.youtube.com/watch?v=5bWG3ySf4fl>

29/02/2024

Panoptic Segmentation

Panoptic segmentation

seeing everything at once (Instance + Semantic segmentation)



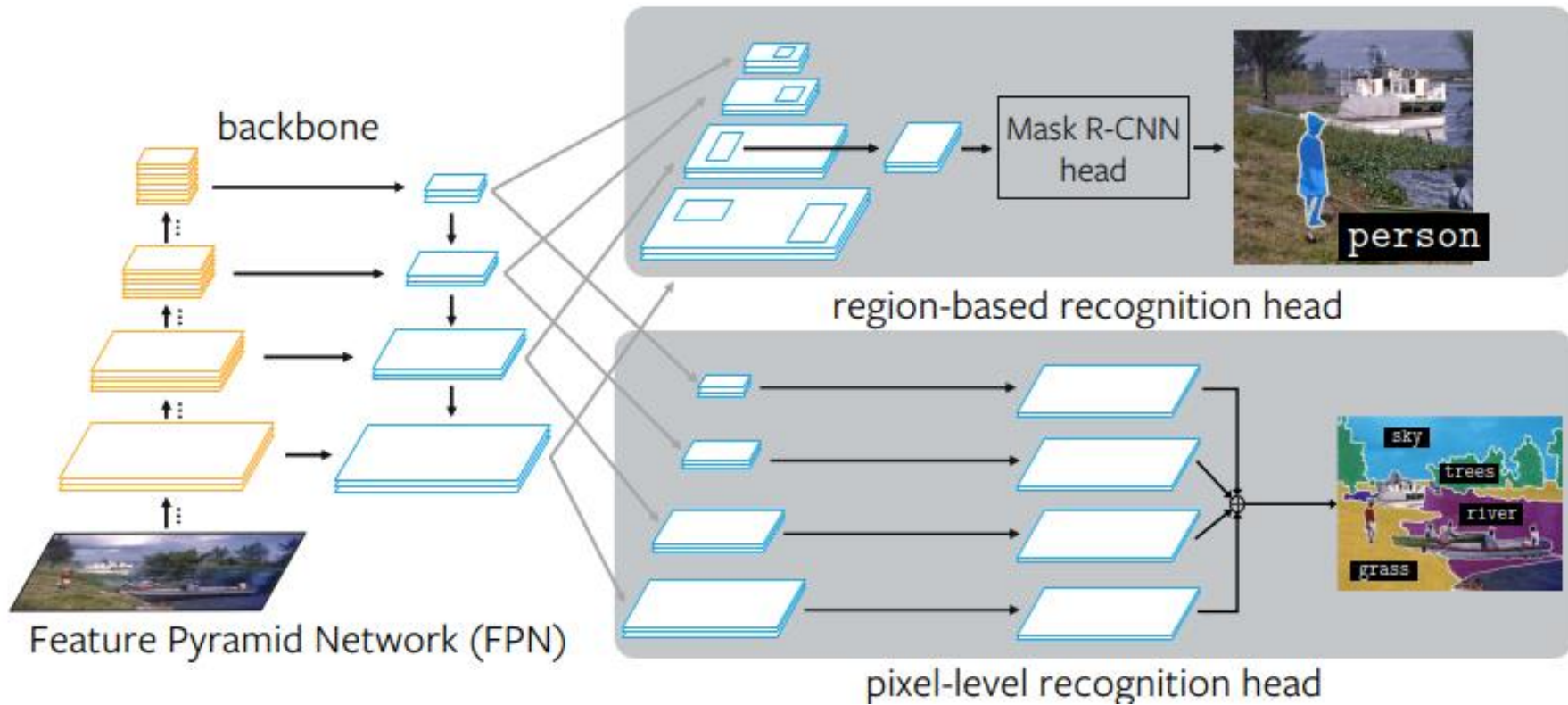
Relevant links:

<https://arxiv.org/abs/1801.00868>

<https://www.youtube.com/watch?v=5bWG3ySf4fl>

Panoptic Segmentation

Panoptic FPN: unified framework



Relevant links:

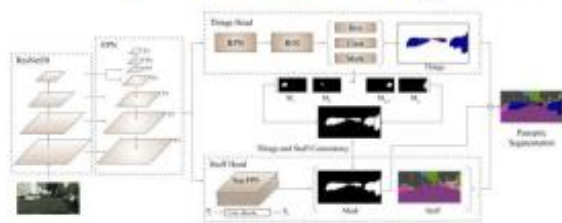
<https://arxiv.org/abs/1801.00868>

<https://www.youtube.com/watch?v=5bWG3ySf4fl>

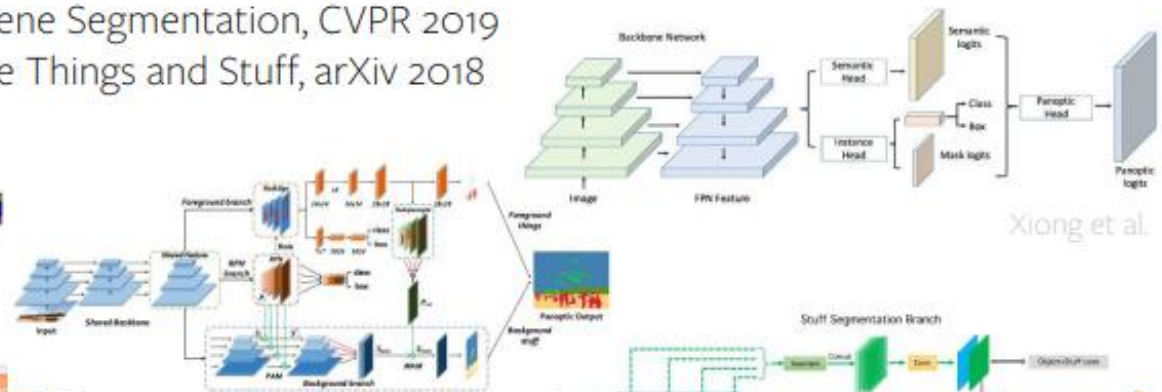
Panoptic Segmentation

Other Implementations:

1. Li et al. Attention-guided Unified Network for Panoptic Segmentation, CVPR 2019
2. Xiong et al. UPSNet: A Unified Panoptic Segmentation Network, CVPR 2019
3. Liu et al. An End-to-End Network for Panoptic Segmentation, CVPR 2019
4. Porzi et al. Seamless Scene Segmentation, CVPR 2019
5. Li et al. Learning to Fuse Things and Stuff, arXiv 2018



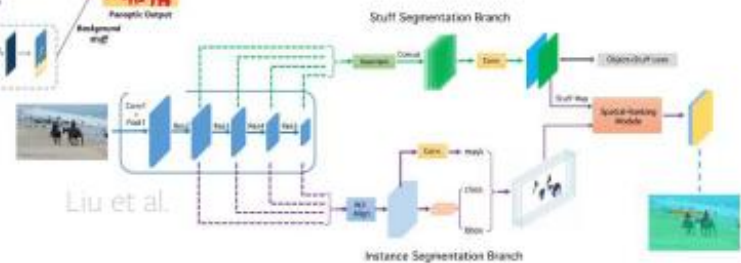
Li et al.



Xiong et al.



Porzi et al.



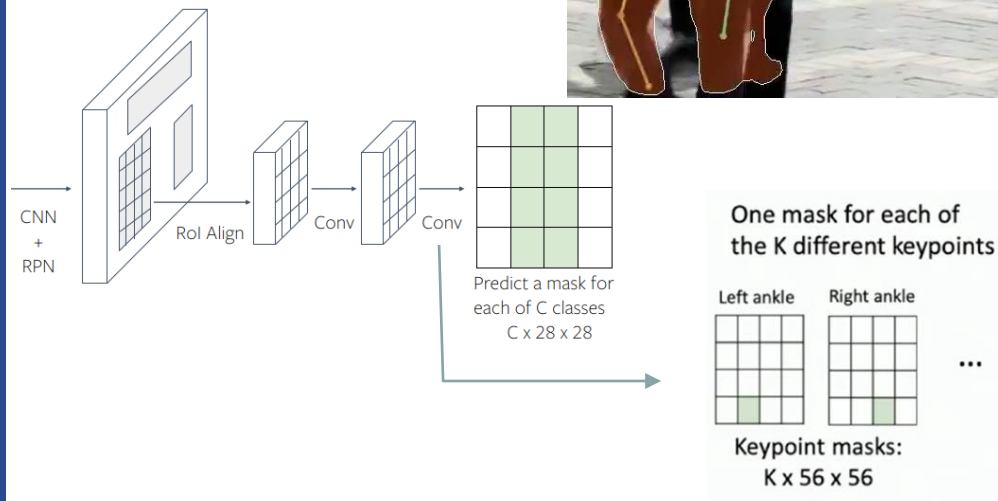
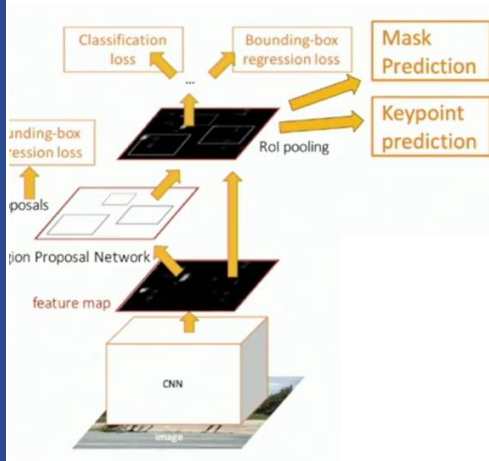
Liu et al.

Schemes credits:
papers on the slide

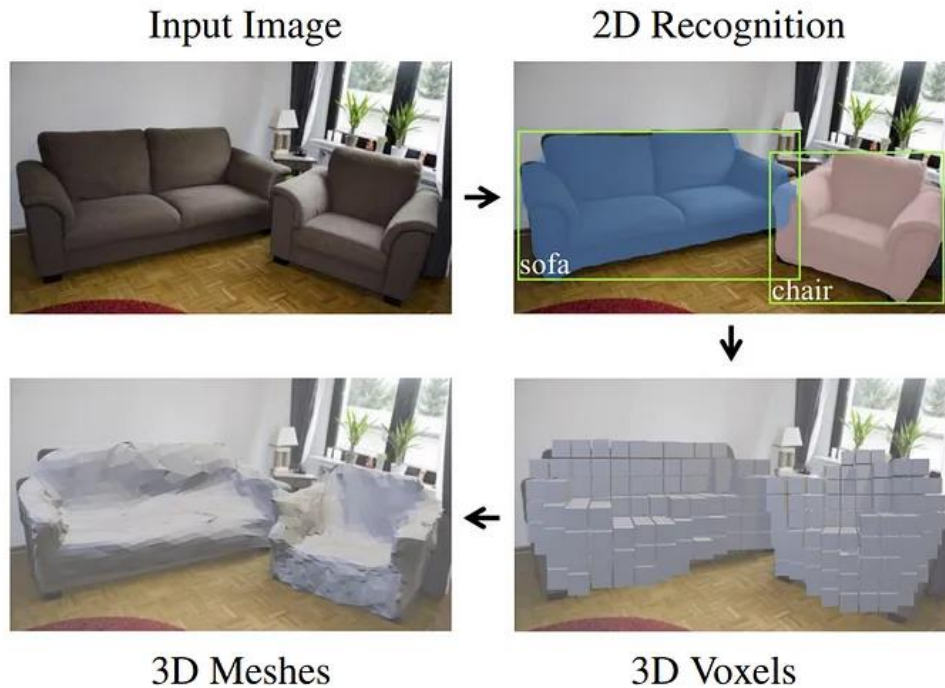
Relevant links:

<https://www.youtube.com/watch?v=5bWG3ySf4fl>

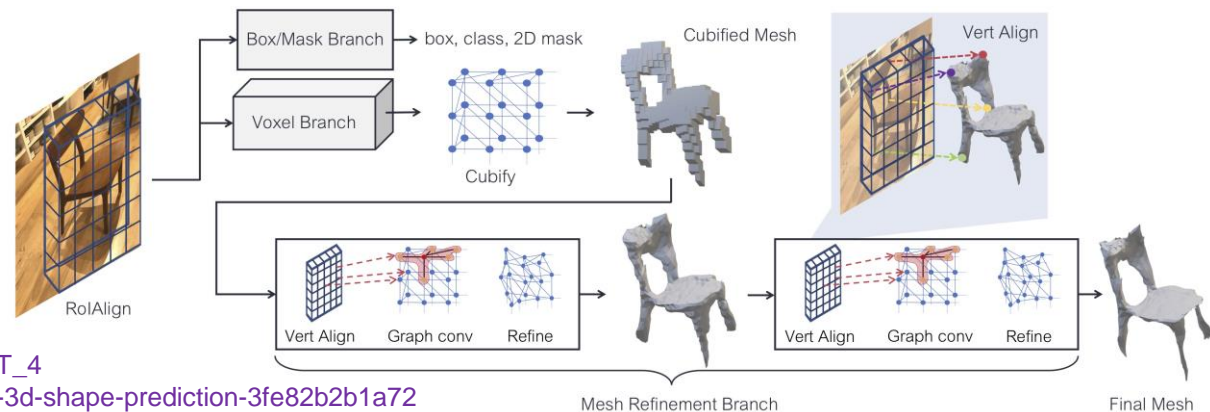
Mask R-CNN and pose estimation



Mesh R-CNN - 3D Shape Prediction



Mesh R-CNN augments Mask R-CNN with a mesh prediction branch that outputs meshes with varying topological structure by first predicting coarse voxel representations which are converted to meshes and refined with a graph convolution network operating over the mesh's vertices and edges.



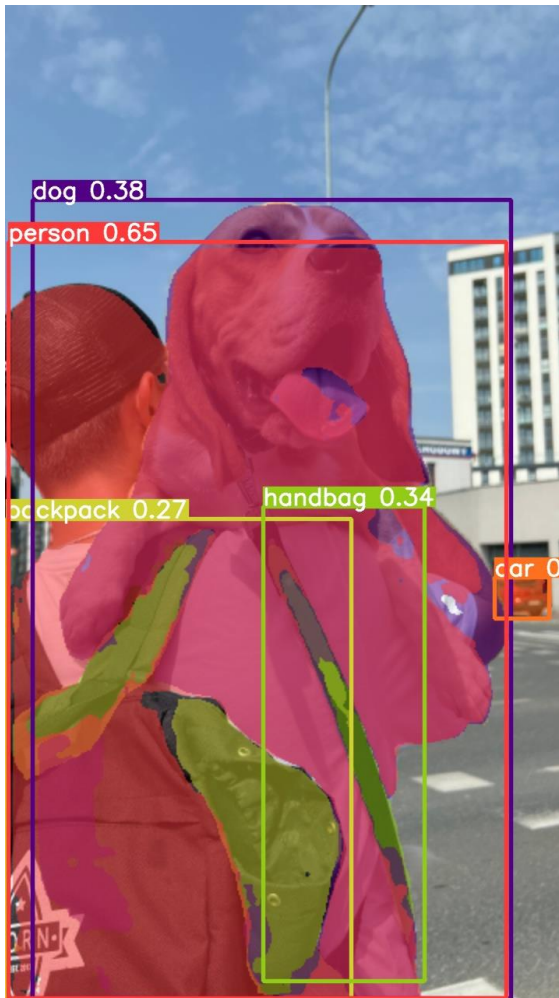
Relevant links:

<https://gkioxari.github.io/meshrcnn/>

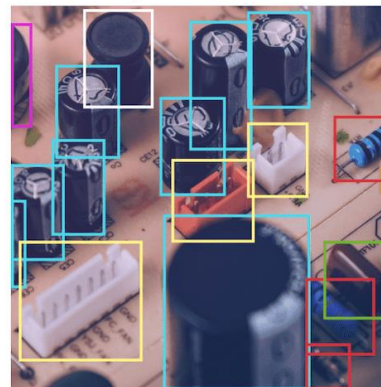
https://www.youtube.com/watch?v=3dNquSS4T_4

<https://medium.com/@davidli2881/mesh-r-cnn-3d-shape-prediction-3fe82b2b1a72>

YOLOv8 - Segmentation

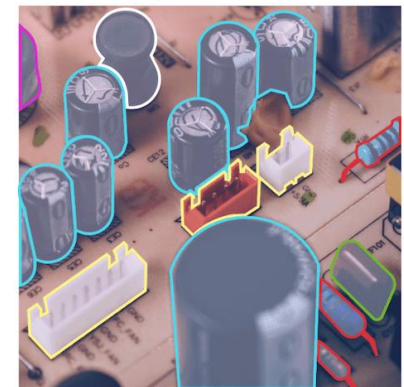


Object Detection



Capacitor, Resistor, Transformer, Connector,
Inductor, Polyester Capacitor

Segmentation



Capacitor, Resistor, Transformer, Connector,
Inductor, Polyester Capacitor

Relevant links:

<https://docs.ultralytics.com/tasks/segmentation/>

<https://www.freecodecamp.org/news/how-to-detect-objects-in-images-using-yolov8/>

<https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolov8-instance-segmentation-on-custom-dataset.ipynb>

YOLOACT (You Only Look At CoefficientTs) is one-stage instance segmentation that is faster, but has lower performance... (2019)

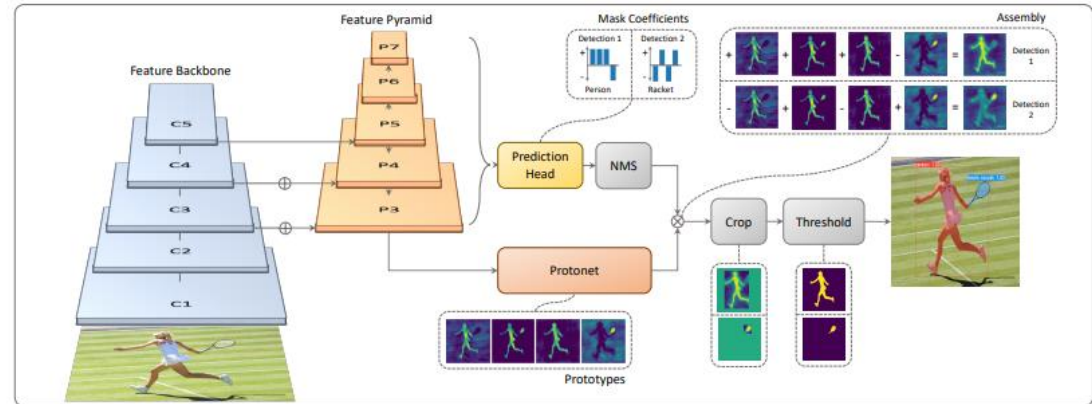
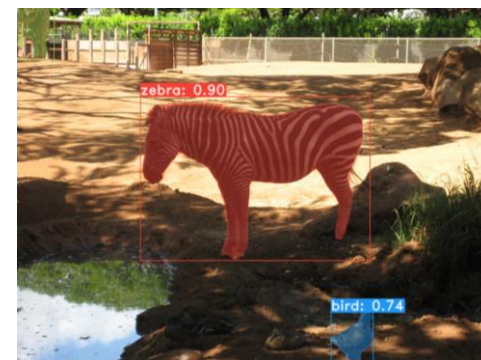
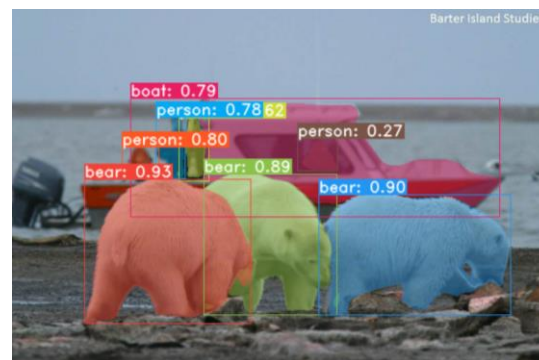


Figure 2: **YOLOACT Architecture** Blue/yellow indicates low/high values in the prototypes, gray nodes indicate functions that are not trained, and $k = 4$ in this example. We base this architecture off of RetinaNet [27] using ResNet-101 + FPN.



Relevant links:

- <https://www.immersivelimit.com/tutorials/yolact-with-google-colab>
- <https://arxiv.org/abs/1904.02689>
- <https://arxiv.org/abs/1912.06218>

29/02/2024

DEtection TRansformer (DETR)

Transformers are a deep learning architecture that has gained popularity... They rely on a simple, yet powerful mechanism called *attention*, which enables AI models to selectively focus on certain parts of their input and thus reason more effectively. Even though originally transformers have been widely applied on problems with sequential data (in particular, in natural language processing (NLP) tasks such as language modeling and machine translation, as well as, extended to tasks of speech recognition, symbolic mathematics, and reinforcement learning), now this approach also shows good results for computer vision tasks like image classification and object detection...



DETR - streamlines the detection pipeline, effectively removing the need for many hand-designed components like a non-maximum suppression procedure or anchor generation that explicitly encode our prior knowledge about the task. The main ingredients of the new framework are a set-based global loss that forces unique predictions via bipartite matching, and a transformer encoder-decoder architecture. Given a fixed small set of learned object queries, DETR reasons about the relations of the objects and the global image context to directly output the final set of predictions in parallel. The new model is conceptually simple and does not require a specialized library, unlike many other modern detectors.

DETR demonstrates accuracy and run-time performance on par with the well-established and highly-optimized Faster RCNN baseline on the challenging COCO object detection dataset. Moreover, DETR can be easily generalized to produce panoptic segmentation in a unified manner. Authors show that it significantly outperforms competitive baselines.

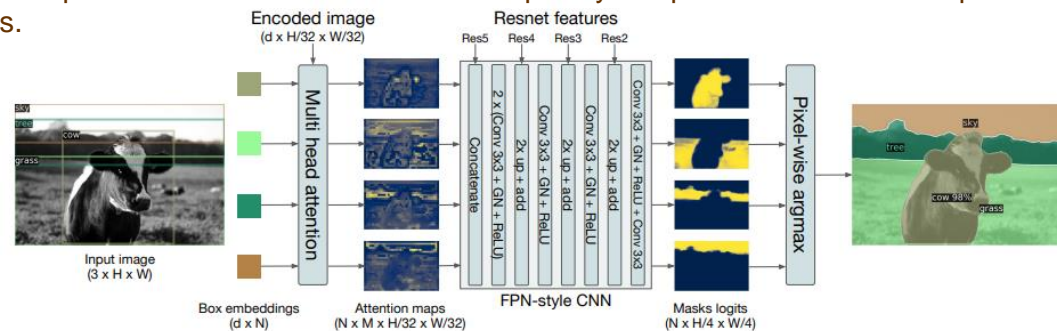


Fig. 8: Illustration of the panoptic head. A binary mask is generated in parallel for each detected object, then the masks are merged using pixel-wise argmax.

Relevant links:

<https://arxiv.org/abs/2005.12872>

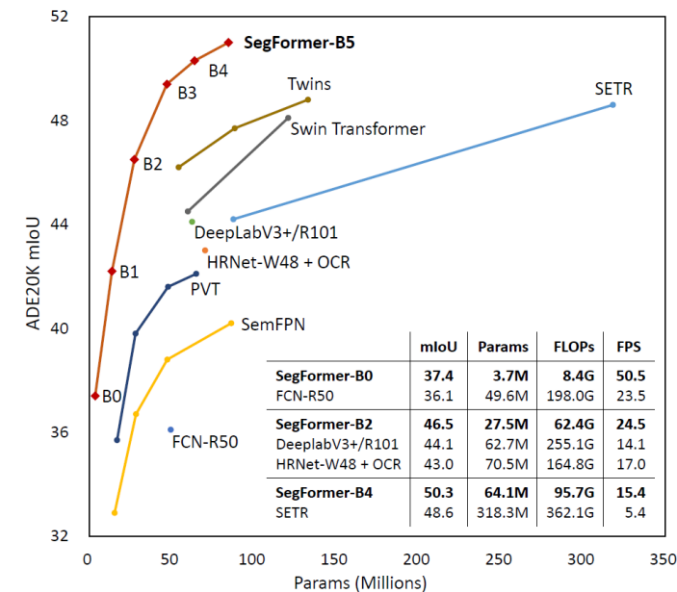
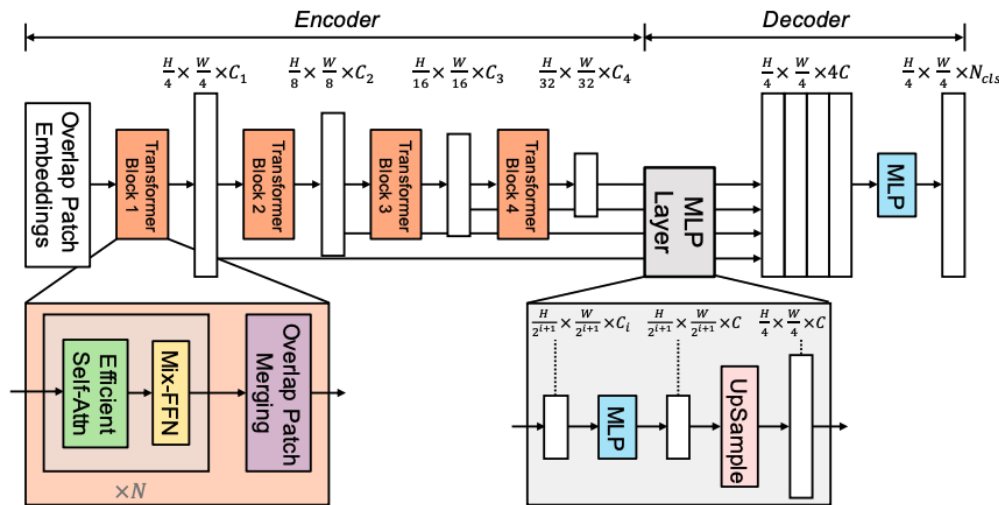
<https://github.com/facebookresearch/detr>

<https://ai.facebook.com/blog/end-to-end-object-detection-with-transformers/>

29/02/2024

SegFormer Simple and Efficient Design for Semantic Segmentation with Transformers. ...

SegFormer is a simple, efficient yet powerful semantic segmentation framework which unifies Transformers with lightweight multilayer perception (MLP) decoders. SegFormer has two appealing features: 1) SegFormer comprises a novel hierarchically structured Transformer encoder which outputs multiscale features. *It does not need positional encoding*, thereby avoiding the interpolation of positional codes which leads to decreased performance when the testing resolution differs from training. 2) SegFormer avoids complex decoders. The proposed MLP decoder aggregates information from different layers, and thus combining both local attention and global attention to render powerful representations. Authors show that this simple and lightweight design is the key to efficient segmentation on Transformers. Approach is scaled up to obtain a series of models from SegFormer-B0 to SegFormer-B5, reaching significantly better performance and efficiency than previous counterparts.



Relevant links:

<https://arxiv.org/abs/2105.15203>

<https://github.com/NVlabs/SegFormer>

<https://keras.io/examples/vision/segformer/>

https://huggingface.co/docs/transformers/en/model_doc/segformer

<https://blog.roboflow.com/how-to-train-segformer-on-a-custom-dataset-with-pytorch-lightning/>

YOLO Object Detection

Object detection with the **YOLO** ...

Darknet is an open-source neural network framework written in C and CUDA made by Joseph Redmon. (<https://pjreddie.com/darknet/>)(<https://github.com/AlexeyAB/darknet>). There is Docker image that contains all the models you need to run Darknet with the following neural networks and models (<https://hub.docker.com/r/loretoparis/darknet/>): **yolo** real time object detection, **imagenet** classification, **nightmare** cnn inception, **rnn** Recurrent neural networks model for text prediction, **darkgo** Go game play

Darkflow is a *Tensorflow* version of *Darknet* made by Th. Thrieu (<https://github.com/thtrieu/darkflow>) that allows you to train and test the YOLO v1 and v2 model.

Alternatively, you may use TensorFlow implementation to use the pre-trained models. Such implementations uses converted models (.ckpt) to be used with TensorFlow or (.h5) for Keras implementations. They also include model convertor (*YOLO weight extractor*).

Implementations:

- <https://github.com/hunglc007/tensorflow-yolov4-tflite>
- <https://github.com/theAIGuysCode/tensorflow-yolov4-tflite>
- <https://neptune.ai/blog/object-detection-with-yolo-hands-on-tutorial>
- <https://www.youtube.com/watch?v=mmj3nxGT2YQ>
- <https://www.youtube.com/watch?v=nOIVxi5yurE>
- <https://reposhub.com/python/deep-learning/david8862-keras-YOLOv3-model-set.html>
- <https://www.youtube.com/watch?v=Grir6TZbc1M>
- <https://github.com/sicara/tf2-yolov4>
- <https://pypi.org/project/yolov4/>
- <https://github.com/qqwweee/keras-yolo3>
- <https://modelzoo.co/model/keras-yolov3>
- <https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/>
- <https://www.youtube.com/watch?v=PyjBd7IDYZs&index=1&list=PLX-LrBk6h3wSGvuTnxB2Kj358XfctL4BM>
- <https://appliedmachinelearning.blog/2018/05/27/running-yolo-v2-for-real-time-object-detection-on-videos-images-via-darkflow/>
- <https://towardsdatascience.com/yolov2-object-detection-using-darkflow-83db6aa5cf5f>
- <https://github.com/experiencor/keras-yolo2>
- <https://www.kaggle.com/sajinpgupta/object-detection-yolov2/notebook>
- <https://github.com/ksanjeewan/dourflow>
- https://github.com/gliese581gg/YOLO_tensorflow
- <https://github.com/nilboy/tensorflow-yolo>
- https://github.com/II_Sourcell/YOLO_Object_Detection
- <https://www.youtube.com/watch?v=lwYUmlIJSvA&list=WL&index=30&t=122s>
- <https://www.youtube.com/watch?v=XRvZuV9RexY>
- https://www.youtube.com/watch?v=b5_B9oNqaqE&list=WL&index=26&t=0s

Relevant links:

<https://github.com/pjreddie/darknet/wiki/YOLO:-Real-Time-Object-Detection>

29/02/2024

YOLO Object Detection

YOLOv3:

https://colab.research.google.com/drive/1Mh2HP_Mfxoao6qNFbhfV3u28tG8jAVGk#scrollTo=HCs4VQmESACK

YOLOv4:

<https://www.youtube.com/watch?v=mmj3nxGT2YQ>

https://colab.research.google.com/drive/1_GdoqCJWXsChrOiY8sZMr_zbr_fH-0Fg?usp=sharing

YOLOX:

<https://github.com/Megvii-BaseDetection/YOLOX>

<https://towardsdatascience.com/how-to-train-yolox-on-a-custom-dataset-bb2f94cdb038>

https://colab.research.google.com/drive/1_xkARB35307P0-BTnqMy0flmYrfoYi5T#scrollTo=igwruhYxE_a7

YOLOR:

<https://viso.ai/deep-learning/yolor/>

<https://github.com/WongKinYiu/yolor>

<https://www.linkedin.com/pulse/how-run-yolor-object-detection-google-colab-ritesh-kanjee/>

YOLOv8:

<https://blog.roboflow.com/how-to-train-yolov8-on-a-custom-dataset/>

<https://keras.io/examples/vision/yolov8/>

https://keras.io/guides/keras_cv/object_detection_keras_cv/

YOLOv9:

<https://github.com/WongKinYiu/yolov9>

<https://docs.ultralytics.com/models/yolov9/>

<https://www.youtube.com/watch?v=rhkYmQ5J3-w>

https://www.youtube.com/watch?v=dccf_sJF0Gg

Tensorflow Object Detection

TensorFlow Object Detection API is an open-source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models.

GitHub: https://github.com/tensorflow/models/tree/master/research/object_detection

TensorFlow 2 meets the OD API: <https://blog.tensorflow.org/2020/07/tensorflow-2-meets-object-detection-api.html>

Object Detection Inference on TF 2 and TF Hub:

https://github.com/tensorflow/hub/blob/master/examples/colab/tf2_object_detection.ipynb

https://www.tensorflow.org/hub/tutorials/tf2_object_detection

<https://www.kaggle.com/models?tfhub-redirect=true>

Related tutorials:

- <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/>
- <https://gilberttanner.com/blog/object-detection-with-tensorflow-2>
- <https://gilberttanner.com/blog/tensorflow-object-detection-with-tensorflow-2-creating-a-custom-model>
- <https://neptune.ai/blog/how-to-train-your-own-object-detector-using-tensorflow-object-detection-api>
- https://www.tensorflow.org/hub/tutorials/object_detection
- https://www.tensorflow.org/tfmodels/vision/object_detection
- <https://www.edureka.co/blog/tensorflow-object-detection-tutorial/>
- <https://pythonprogramming.net/introduction-use-tensorflow-object-detection-api-tutorial/>
- <https://github.com/EdgeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10>
- <https://www.curiously.com/posts/object-detection-on-custom-dataset-with-tensorflow-2-and-keras-using-python/>
- https://github.com/pythonlessons/TensorFlow-object-detection-tutorial/tree/master/5_part%20step%20by%20step%20custom%20object%20detection
- https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.md
- https://github.com/tensorflow/models/blob/master/research/object_detection/object_detection_tutorial.ipynb
- <https://www.oreilly.com/ideas/object-detection-with-tensorflow>
- <https://medium.com/@WuStangDan/step-by-step-tensorflow-object-detection-api-tutorial-part-1-selecting-a-model-a02b6aabe39e>

Creating an Object Detection Application on Google Cloud using TensorFlow

(tutorial with billable components): <https://cloud.google.com/solutions/creating-object-detection-application-tensorflow>

TensorFlow Object Detection with Docker from scratch (tutorial)

<https://medium.com/@evheniybystrov/tensorflow-object-detection-with-docker-from-scratch-5e015b639b0b>

Object Detection

DEtection TRansformer (DETR):

<https://github.com/facebookresearch/detr>

https://colab.research.google.com/github/facebookresearch/detr/blob/colab/notebooks/detr_attention.ipynb

<https://www.geeksforgeeks.org/object-detection-with-detection-transformer-detr-by-facebook/>

<https://huggingface.co/facebook/detr-resnet-50-panoptic>

Object detection with Vision Transformer (ViT):

https://keras.io/examples/vision/object_detection_using_vision_transformer/

RetinaNet:

<https://medium.com/@van.evanfebrianto/how-to-train-custom-object-detection-models-using-retinanet-aeed72f5d701>

<https://github.com/fizyr/keras-retinanet>

<https://cloud.google.com/tpu/docs/tutorials/retinanet-2.x>

<https://keras.io/examples/vision/retinanet/>

FCOS:

<https://github.com/tianzhi0549/FCOS>

CenterNet:

<https://colab.research.google.com/github/sony/nnabla-examples/blob/master/interactive-demos/centernet.ipynb>

<https://github.com/sony/nnabla-examples/tree/master/object-detection/centernet>

Papers with code:

<https://paperswithcode.com/task/object-detection>

Image Segmentation

U-Net:

- https://keras.io/examples/vision/oxford_pets_image_segmentation/

MaskRCNN:

- https://github.com/tensorflow/hub/blob/master/examples/colab/tf2_object_detection.ipynb
- <https://gilberttanner.com/blog/train-a-mask-r-cnn-model-with-the-tensorflow-object-detection-api>
- https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/instance_segmentation.md
- https://github.com/matterport/Mask_RCNN
- https://github.com/markjay4k/Mask-RCNN-series/blob/master/Mask_RCNN%20Install%20Instructions.ipynb
- <https://gilberttanner.com/blog/getting-started-with-mask-rcnn-in-keras>
- <https://github.com/ahmedfgad/Mask-RCNN-TF2>
- <https://www.immersivelimit.com/tutorials/mask-rcnn-for-windows-10-tensorflow-2-cuda-101>

YOLOACT:

- https://colab.research.google.com/drive/1ncRxvmNR-iTtQCscj2UFSGV8ZQX_LN0M

YOLOv8:

- <https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolov8-instance-segmentation-on-custom-dataset.ipynb#scrollTo=AFMBYQtMVL-B>

DeepLab v3:

- <https://github.com/bonlime/keras-deeplab-v3-plus> , https://keras.io/examples/vision/deeplabv3_plus/

Segmentation Models:

- https://github.com/bonlime/segmentation_models

Segmentation with TF:

- <https://www.tensorflow.org/tutorials/images/segmentation>
- https://www.tensorflow.org/tfmodels/vision/semantic_segmentation
- https://www.tensorflow.org/tfmodels/vision/instance_segmentation
- https://github.com/tensorflow/models/blob/master/samples/outreach/blogs/segmentation_blogpost/image_segmentation.ipynb
- <https://ai.googleblog.com/2018/03/semantic-image-segmentation-with.html>
- <https://medium.freecodecamp.org/how-to-use-deeplab-in-tensorflow-for-object-segmentation-using-deep-learning-a5777290ab6b>
- <https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef>

Segmentation with Fully Convolutional Networks (FCN):

- https://keras.io/examples/vision/fully_convolutional_network/

Hugging Face Image Segmentation Models: https://huggingface.co/models?pipeline_tag=image-segmentation



Image Segmentation

Segmentation with Transformers:

- <https://keras.io/examples/vision/segformer/>
- <https://keras.io/examples/vision/sam/>
- <https://blog.roboflow.com/how-to-train-segformer-on-a-custom-dataset-with-pytorch-lightning/>

Hugging Face Image Segmentation Models: https://huggingface.co/models?pipeline_tag=image-segmentation

Systems and Tools

Similarly to Google, in January 2018, Facebook has come out with a platform for object detection of it's own - **Detectron**.

Detectron was written in Python and leveraged the Caffe2 deep learning framework underneath. <https://research.fb.com/facebook-open-sources-detectron/>, <https://github.com/facebookresearch/Detectron>

Initial version of Detectron is currently deprecated in favor of a second version **Detectron2**. It is a ground-up rewrite of Detectron in PyTorch, and originates from maskrcnn-benchmark.

<https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library/>

<https://github.com/facebookresearch/detectron2>

<https://ai.facebook.com/tools/detectron2/>

The goal of **Detectron** is to provide a high-quality, high-performance codebase for object detection research. It is designed to be flexible in order to support rapid implementation and evaluation of novel research. **Detectron2** includes high-quality implementations of state-of-the-art object detection algorithms, including DensePose, panoptic feature pyramid networks, and numerous variants of the pioneering Mask R-CNN model family also developed by FAIR.

Relevant links:

<https://www.dlology.com/blog/how-to-train-detectron2-with-custom-coco-datasets/>

https://colab.research.google.com/github/styler00dollar/Colab-Detectron2/blob/resnest/Colab-Detectron2.ipynb#scrollTo=9_FzH13EjseR

https://colab.research.google.com/drive/16jcaJoc6bCFAQ96jDe2HwtXj7BMD_-m5

<https://colab.research.google.com/drive/1-TNOcPm3Jr3fOJG8rnGT9gh60mHUsvaW>

<https://medium.com/@mdagdalen/detectron2-training-on-custom-coco-dataset-b0e8d4b6b3b>

<https://towardsdatascience.com/a-beginners-guide-to-object-detection-and-computer-vision-with-facebook-s-detectron2-700b6273390e>

<https://www.analyticsvidhya.com/blog/2018/01/facebook-launched-detectron-platform-object-detection-research/>

<https://hackernoon.com/how-to-use-detectron-facebooks-free-platform-for-object-detection-9d41e170bbcb>

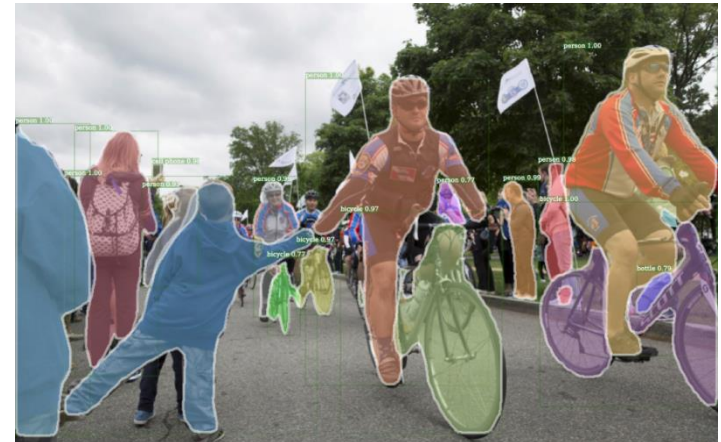
<https://arxiv.org/abs/1703.06870/>; <https://arxiv.org/abs/1708.02002>; <https://arxiv.org/abs/1506.01497>; <https://arxiv.org/abs/1506.01497>;

<https://arxiv.org/abs/1504.08083>; <https://arxiv.org/abs/1605.06409>; <https://arxiv.org/abs/1611.05431>; <https://arxiv.org/abs/1512.03385>;

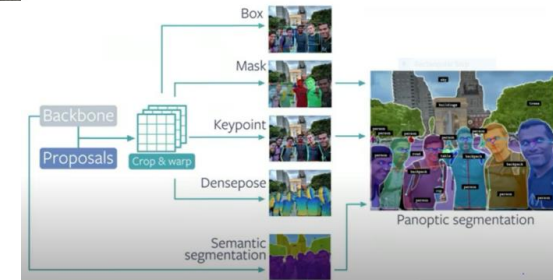
<https://arxiv.org/abs/1612.03144>; <https://arxiv.org/abs/1409.1556>;

29/02/2024

TIES4911 – Lecture 6



- Mask R-CNN
- RetinaNet
- Faster R-CNN
- RPN
- Fast R-CNN
- TensorMask
- PointRend
- DensePose
- ...



Systems and Tools

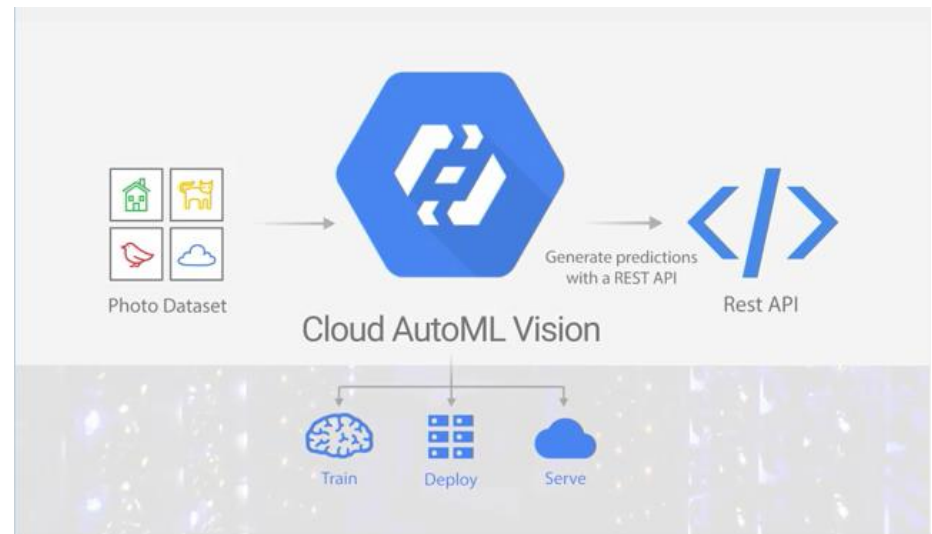
January 2018, Google announced the launch of **Cloud AutoML platform** to help businesses with limited resources and expertise, streamline and build high-quality machine learning models.

Google believes this will provide opportunities for less-skilled engineers to build powerful AI systems and make AI experts even more productive and efficient.

Cloud AutoML Vision – is a first product, as part of the Cloud AutoML portfolio, to make it simpler to train image recognition models. It has a drag-and-drop interface that let's the user upload images, train the model, and then deploy those models directly on Google Cloud. Cloud AutoML Vision is built on Google's *transfer learning* and *neural architecture search* technologies (among others).

<https://cloud.google.com/automl/>

<https://www.youtube.com/watch?v=GbLQE2C181U>



The entire *Cloud AutoML platform* will help businesses scale up their AI capabilities without requiring advanced machine learning expertise. It opens the door for non-technical people to understand and utilize machine learning to improve not only their portfolio, but their company's as well. Google has promised more products under the Cloud AutoML umbrella so those who don't like coding can enjoy this latest renaissance in the ML and AI fields.

Relevant links:

<https://www.blog.google/topics/google-cloud/cloud-automl-making-ai-accessible-every-business/>

<https://www.analyticsvidhya.com/blog/2018/01/google-cloud-automl-platform-build-models-without-coding/>

Systems and Tools

IBM PowerAI Vision makes computer vision with deep learning more accessible to business users. PowerAI Vision includes an intuitive toolset that empowers subject matter experts to label, train, and deploy deep learning vision models, without coding or deep learning expertise.

<https://www.ibm.com/fi-en/marketplace/ibm-powerai-vision>

Supervisely the leading platform for entire computer vision lifecycle. Image annotation and data management tools transform your images / videos / 3d point cloud into high-quality training data. Train your models, track experiments, visualize and continuously improve model predictions, build custom solution within the single environment.

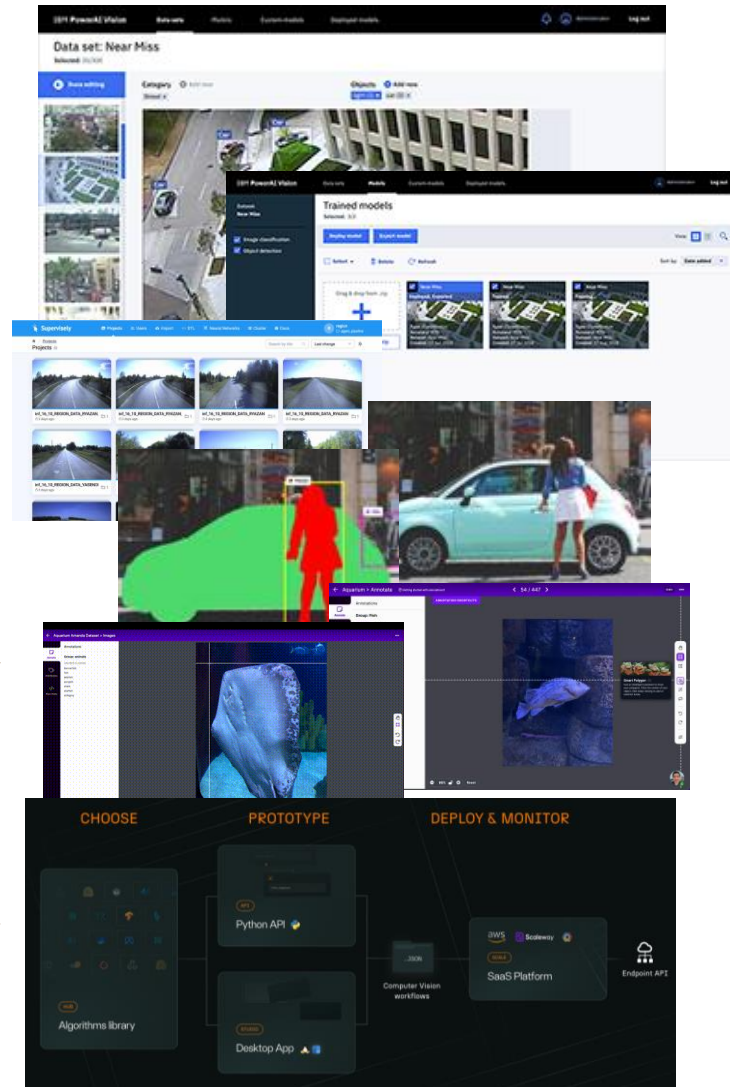
<https://supervise.ly>
https://www.youtube.com/watch?v=7eOLm_dsuz8&t=69s

Roboflow Annotate is a self-serve annotation tool included with all Roboflow accounts that greatly streamlines the process of going from raw images to a trained and deployed computer vision model. Whether you need to correct a single annotation or label an entire dataset, you can now do it within Roboflow without having to download a separate program or go through the process of exporting and re-importing your images.

<https://docs.roboflow.com/annotate>

ikomia is an open-source framework to build and deploy Computer Vision solutions without the hassle.

<https://www.ikomia.ai/>
<https://github.com/ikomia-dev>




Systems and Tools



ImageAI a python library built to empower developers to build applications and systems with self-contained Deep Learning and Computer Vision capabilities using simple and few lines of code. Built with simplicity in mind, ImageAI supports a list of state-of-the-art Machine Learning algorithms for image prediction, custom image prediction, object detection, video detection, video object tracking and image predictions trainings.

<https://imageai.readthedocs.io/en/latest/#>
<https://github.com/OlafenwaMoses/ImageAI>

 **GluonCV** is one of the best library frameworks with most of the state-of-the-art implementations for deep learning algorithms for various computer vision applications.

<https://cv.gluon.ai/>
<https://github.com/dmlc/gluon-cv>

 **MediaPipe** The **MediaPipe Object Detector** lets you detect the presence and location of multiple classes of objects.

https://developers.google.com/mediapipe/solutions/vision/object_detector/python
https://mediapipe-studio.webapps.google.com/demo/object_detector

fast.ai **fast.ai** is a deep learning library which provides practitioners with high-level components that can quickly and easily provide state-of-the-art results in standard deep learning domains and provides researchers with low-level components that can be mixed and matched to build new approaches.

<https://www.fast.ai/posts/2020-02-13-fastai-A-Layered-API-for-Deep-Learning.html>
<https://medium.com/@c.kuan/how-to-run-fast-ai-on-google-colab-25c5a19ea616>

Systems and Tools

V7 is the AI Data Engine to solve any labeling task 10x faster for computer vision and generative AI.

<https://www.v7labs.com/>

https://storage.googleapis.com/openimages/web/factsfigures_v7.html

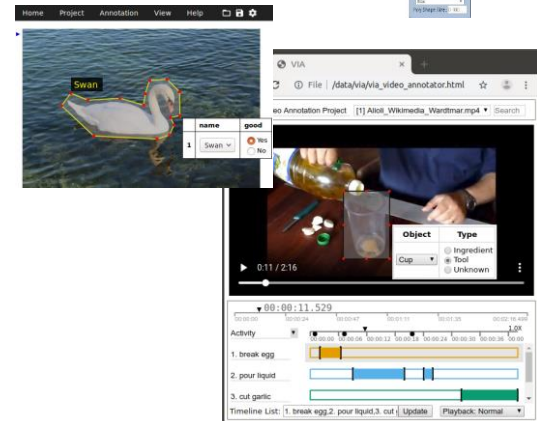
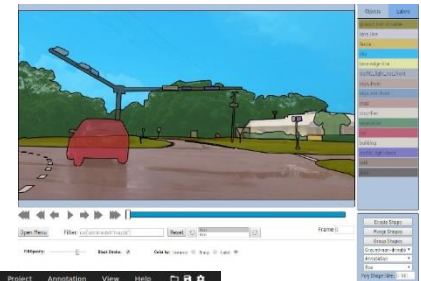
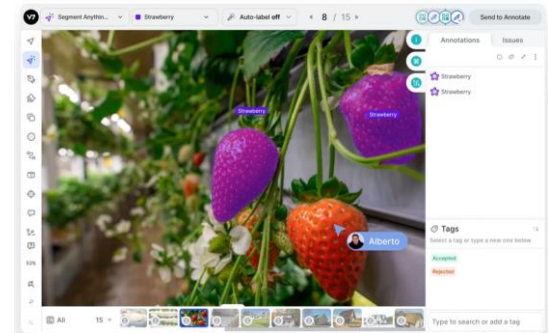
Computer Vision Annotation Tool (CVAT) is free, online, interactive video and image annotation tool for computer vision. It is being used to annotate million of objects with different properties. Many UI and UX decisions are based on feedbacks from professional data annotation team.

<https://github.com/opencv/cvat>

<https://software.intel.com/en-us/articles/computer-vision-annotation-tool-a-universal-approach-to-data-annotation>

VGG Image Annotator (VIA) is a simple and standalone manual annotation software for image, audio and video. VIA runs in a web browser and does not require any installation or setup. The complete VIA software fits in a single self-contained HTML page of size less than 400 Kilobyte that runs as an offline application in most modern web browsers.

<http://www.robots.ox.ac.uk/~vgg/software/via/>



Relevant links:

<https://lionbridge.ai/articles/image-annotation-tools-for-computer-vision/>

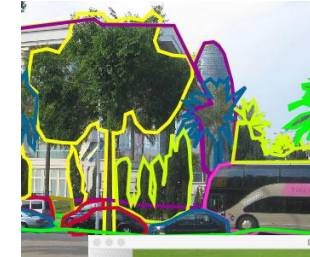
Systems and Tools

LabelImg is a graphical image annotation tool. It is written in Python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format, the format used by ImageNet. Besides, it also supports YOLO format. <https://github.com/tzutalin/labelImg>

LabelMe is an online annotation tool to build image databases for computer vision research. <http://labelme.csail.mit.edu/Release3.0/>

RectLabel an image annotation tool to label images for bounding box object detection and segmentation. <https://rectlabel.com/>

COCO Annotation UI project includes the front end user interfaces that are used to annotate COCO dataset. For more details, please visit COCO. <https://github.com/tylin/coco-ui>

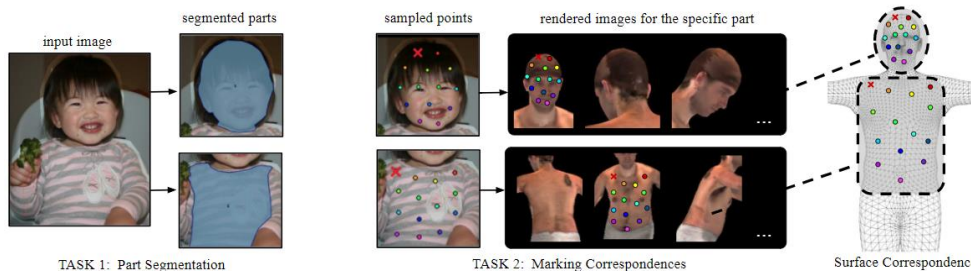
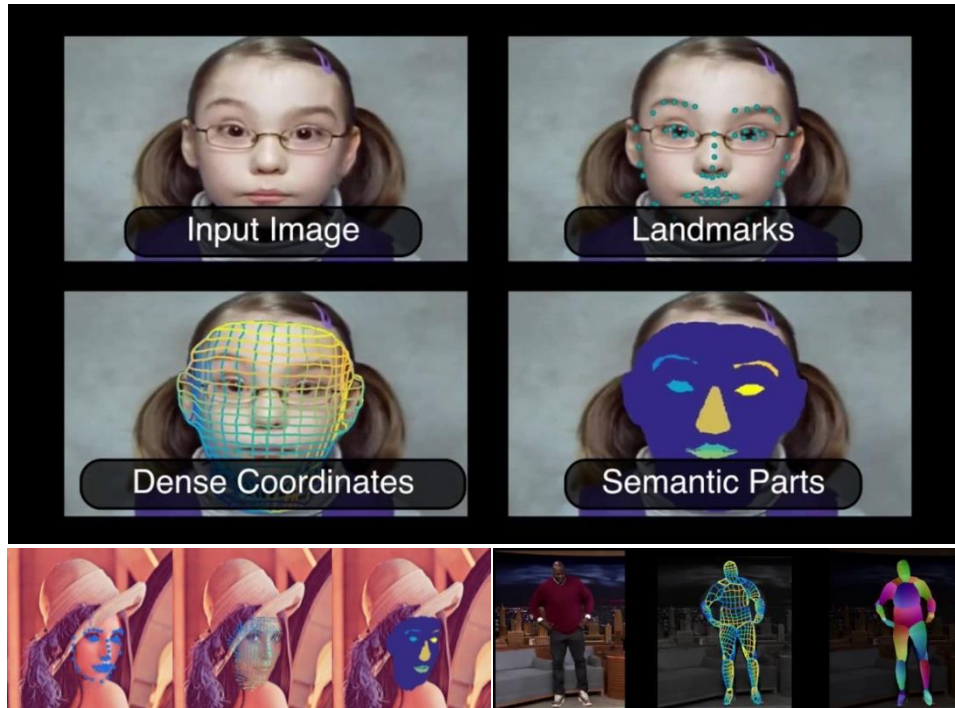


Relevant links:

<https://lionbridge.ai/articles/image-annotation-tools-for-computer-vision/>

Computer Vision

Face (and body pose) analysis with DenseReg



Guler et.al. CVPR 2017: DenseReg: Fully Convolutional Dense Shape Regression In-the-Wild
<https://arxiv.org/pdf/1612.01202.pdf>

Web: <http://alpguler.com/DenseReg.html>

Video: https://www.youtube.com/watch?v=RZbQXpQ_wdg

Code: <https://github.com/ralpguler/DenseReg>

Guler et.al. submitted Feb 2018: DensePose: Dense Human Pose Estimation In The Wild

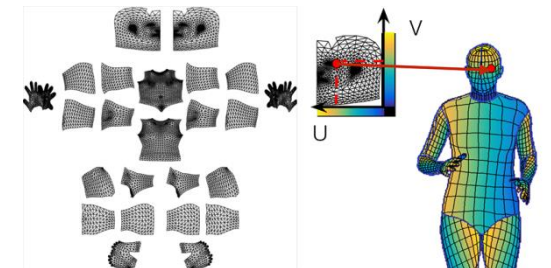
<https://arxiv.org/abs/1802.00434>

<http://densepose.org/>

<https://github.com/facebookresearch/DensePose>

<https://medium.com/@kunalchhabria/overview-of-densepose-dense-human-pose-estimation-in-the-wild-67b0eb7c508a>

DensePose-RCNN System





Neural Style Transfer

Deep Learning is more than just classification or regression...

Object recognition models (e.g. VGG) are invariant to different object representations (style, shape, background, etc.).
Therefore, *content and style can be separated*.

Relevant links:

<https://arxiv.org/abs/1705.04058>

<https://arxiv.org/abs/1505.07376>

<https://arxiv.org/abs/1508.06576>

<https://github.com/jcjohnson/neural-style>

<https://github.com/jcjohnson/fast-neural-style>

29/02/2024

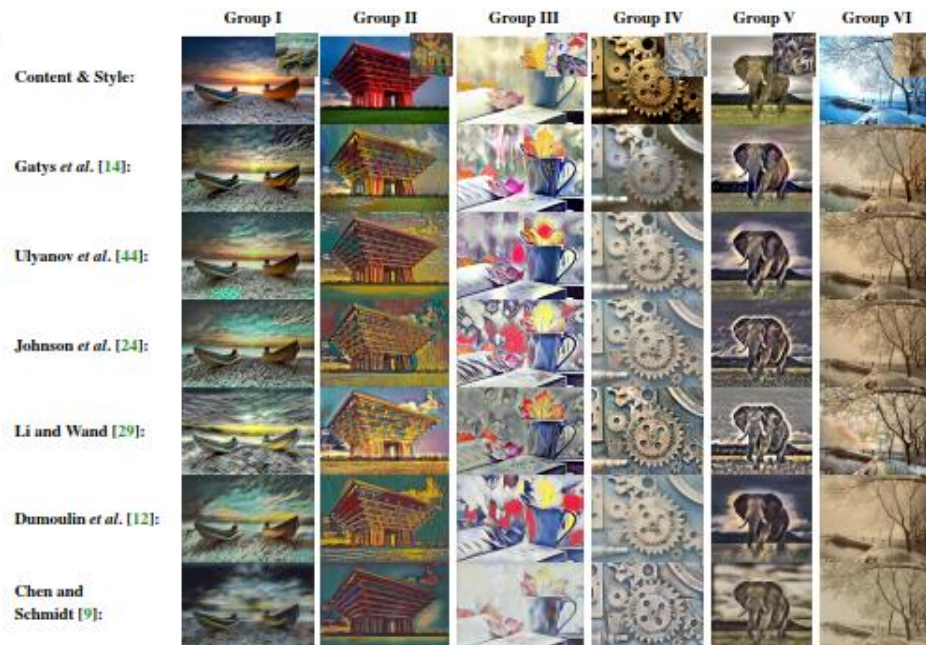
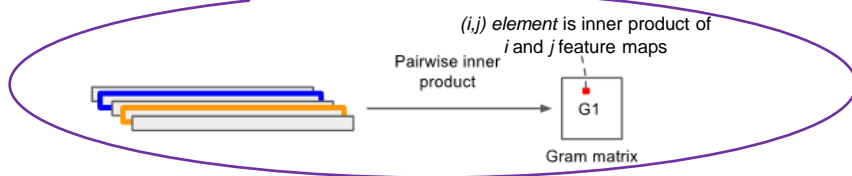
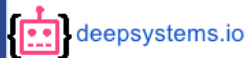
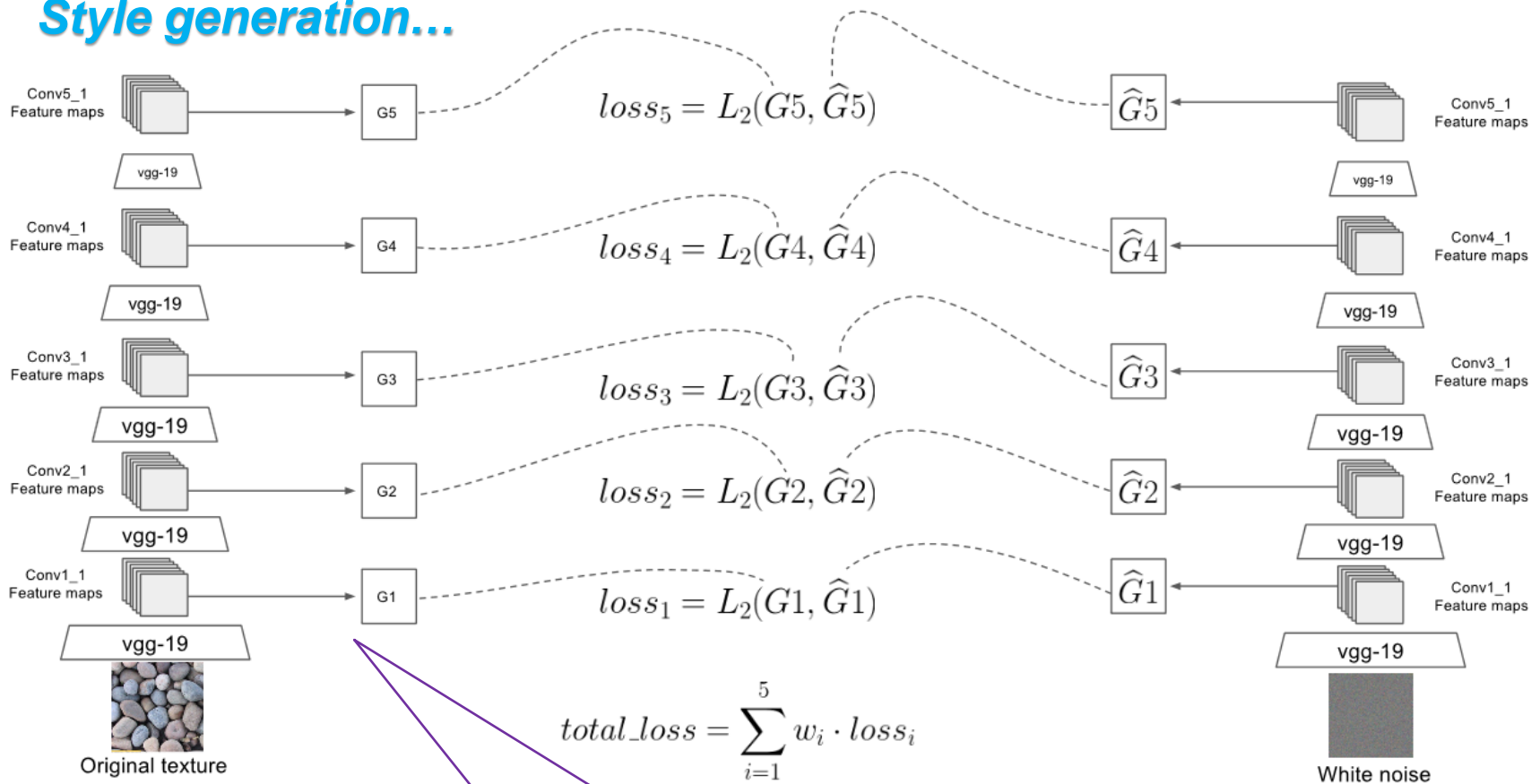


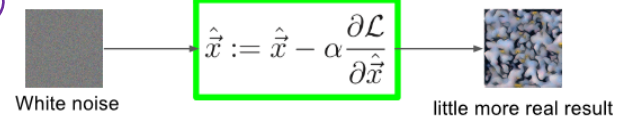
Figure 3. Some example results for qualitative evaluation.

Neural Style Transfer

Style generation...



Update image pixels after backpropagation...

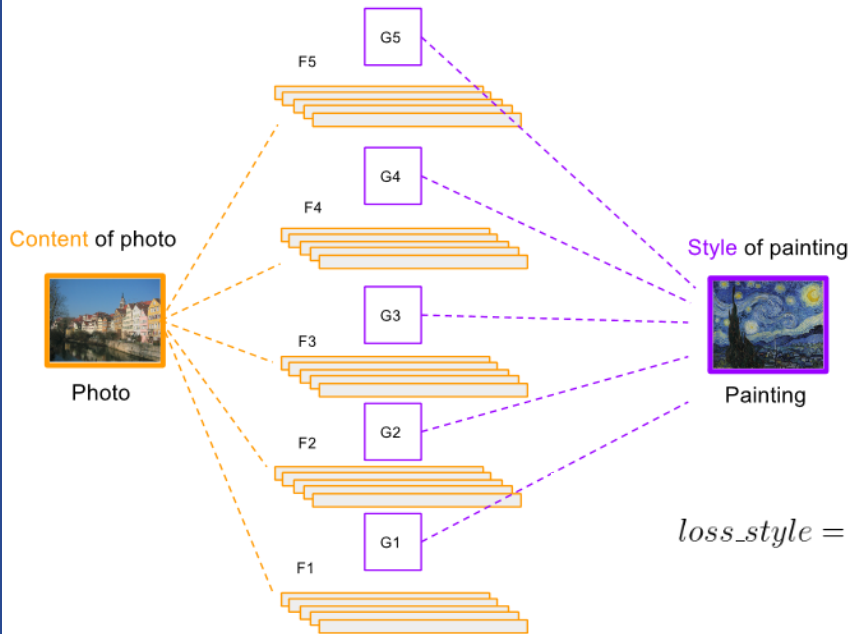


Relevant links:
<https://deepsystems.ai>

29/02/2024

Neural Style Transfer

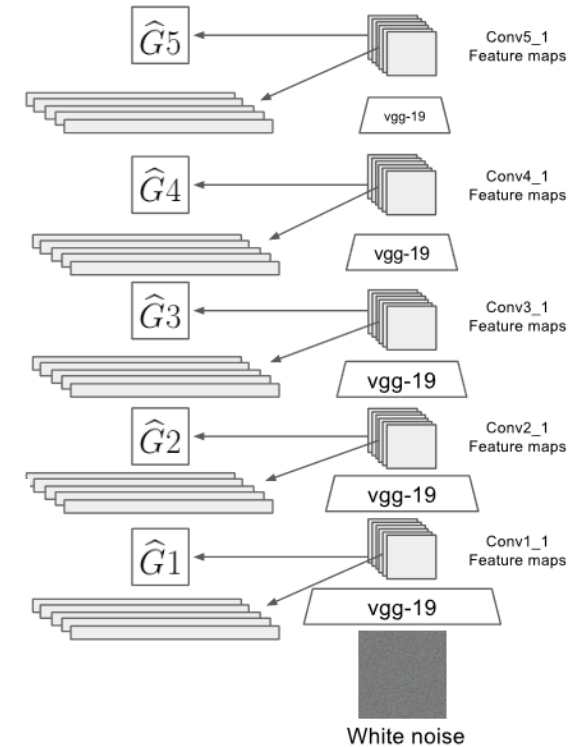
Style transfer...



$$loss_style = \sum_{i=1}^5 w_{s_i} \cdot L_2(G_i, \hat{G}_i)$$

$$loss_content = \sum_{i=1}^5 w_{c_i} \cdot L_2(F_i, \hat{F}_i)$$

$$total_loss = \alpha \cdot loss_content + \beta \cdot loss_style$$

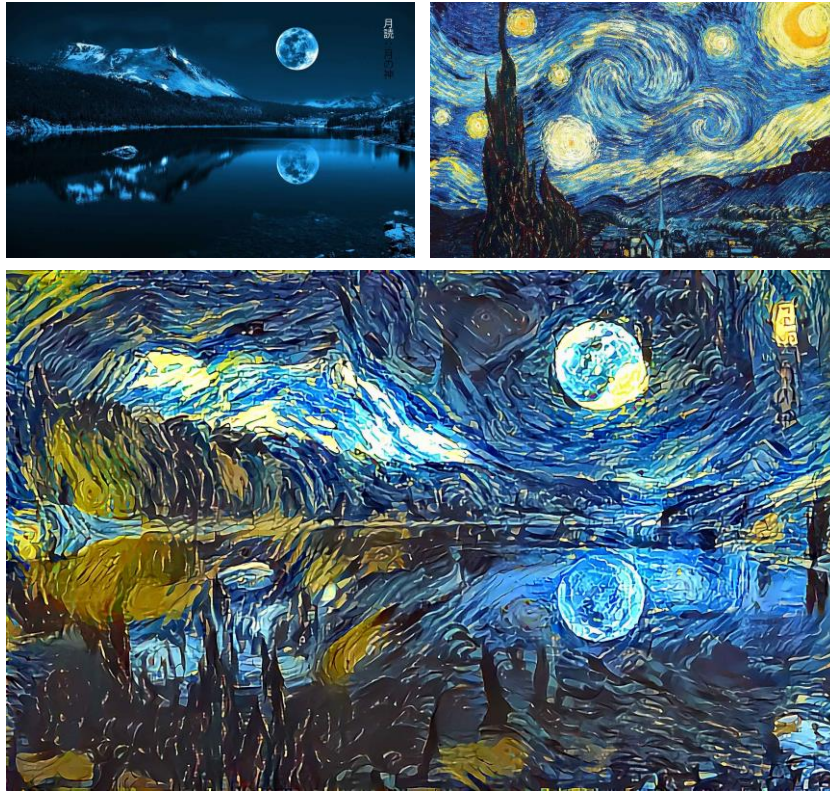


Relevant links:

<https://deepsystems.ai>

29/02/2024





Neural Style Transfer



Demo:

- <https://reiinakano.com/arbitrary-image-stylization-tfjs/>
- <https://huggingface.co/spaces/aravinds1811/neural-style-transfer>

Tutorials and Implementations:

- https://www.tensorflow.org/tutorials/generative/style_transfer
- https://www.tensorflow.org/hub/tutorials/tf2_arbitrary_image_stylization
- <https://www.datacamp.com/community/tutorials/implementing-neural-style-transfer-using-tensorflow>
- https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf
- https://github.com/tensorflow/models/blob/master/research/nst_blogpost/4_Neural_Style_Transfer_with_Eager_Execution.ipynb
- <https://github.com/titu1994/Neural-Style-Transfer>
- <https://github.com/anishathalye/neural-style>
- <https://github.com/cysmith/neural-style-tf>
- <https://github.com/lengstrom/fast-style-transfer>
- <https://www.v7labs.com/blog/neural-style-transfer> , <https://harishnarayanan.org/writing/artistic-style-transfer/>