FINAL STUDY REPORT

Ares – Kaicheng Jia

June 2020

1 CONTENT

- *Week 5:*
    1. *Edge Machine Learning*
        - *Intel Movidius*
            - *Ubuntu VM*
            - *Neural Compute Application Zoo*
        - *iOS CoreML*
            - *MXNet Convertor*
            - *TensorFlow Convertor*
        - *AWS DeepLense*
            - *Sample Project*
            - *New Project*
        - *AWS SageMaker*
    2. *TensorFlow Model Optimization Tookit*
        - *Weight Pruning (with Keras)*
        - *TensorFlow Lite*
        - *Post-training quantization*
    3. *TensorFlow Lite*
    4. *Hyperparameter Tuning*
        - *Grid Search & Random Search*
        - *HParams Dashboard (grid search)*
        - *Keras Tuner (random search)*
    5. *Neural Architecture Search (NAS)*
        - *Search Space*
        - *Search Strategy*
        - *Performance Estimation Strategy*
    6. *Other Materials*
        - *Pyimagesearch, Fast.ai, Introtodeeplearning.com*

- *Week 4:*
    1. *AutoML, AutoML Vision*
        - *AWS Machine Learning*
        - *AWS SageMaker Studio*
        - *Google Cloud Platform*
        - *LUDWIG*
        - *H2O*
    2. *Object Detection*
        - *Region Proposal*
        - *Selective Search*
        - *Feature Extraction*
        - *Evaluation Metric： IoU& mAP*
        - *Precision & Recall*
    3. *R-CNN*
    4. *Faster R-CNN*
    5. *Amazon Rekogonition API*
    6. *Data Scramping - fake-useragent*
- *Week 3:*
    1. *AWS Cloud 9*
    2. *cGAN*
    3. *cycleGAN*
    4. *APIs*
- *Week 2:*
    1. *Deep Reinforcement Learning and GANs*
    2. *TensorFlow playground & TensorFlow Hub*
    3. *Generative Modeling*
    4. *Transfer Learning*
    5. *Open Gym*
    6. *AWS DeepRacer*
    7. *Reinforcement, Supervised, Unsupervised Learning*
    8. *Autoencoder*
    9. *Variational Autoencoder (VAE)*
    10. *Generative Adversarial Network*
    11. *Feature Engineering*
- *Week 1:*
    1. *Computer Vision Cases and Challenges*
    2. *Edge Based Devices*
    3. *CPU, GPU, FPGA, TPU*

4. *Emerging hardware and AI and managed ML platforms*
5. *TensorFlow Russian Dolls*
6. *Biological & Machine Vision*
7. *Colab & Jupyter Notebook*
8. *Cloud AI Development with AWS*
9. *CNN – LeNet5 – AlexNet – DNN – Transfer Learning*
10. *Dataset: MNIST, CIFAR10*

2 NOTES

Week 5:

- *Intel Movidius*
  - *We use Ubuntu VM as a separate operating system for convenience, and it's the most straightforward path to install SDK*
  - *Neural Compute Application Zoo:* https://github.com/movidius/ncappzoo
- *iOS CoreML*
  - *Optimized for on-device performance*
  - *Minimal memory footprint*
  - *Can convert trained models from third-party machine learning framework and we don't even have to train the model:*

```
pip install -U coremltools
```

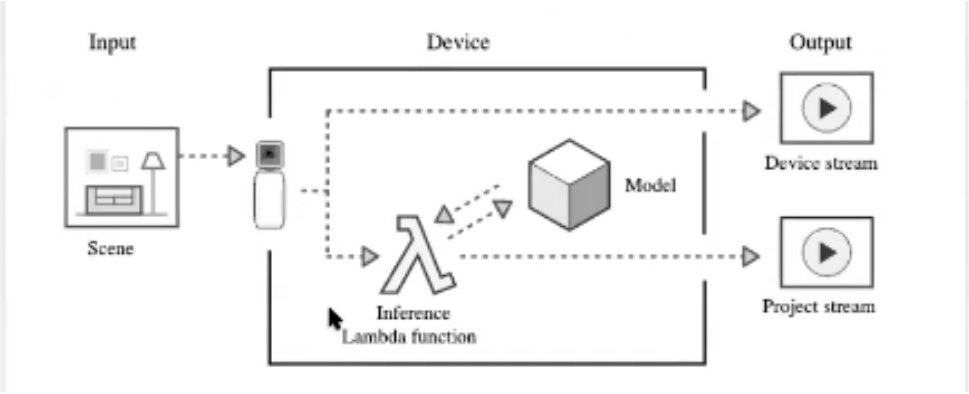
```
# import coremltools
import coremltools

# convert model
coreml_model = coremltools.converters.caffe.convert('my_caffe_model.caffemodel')

# save model to Core ML model format
coremltools.utils.save_spec(coreml_model, 'my_model.mlmodel')
```

  - *Examples: CoreML Image Classification* https://developer.apple.com/documentation/vision/classifying_images_with_vision_and_core_ml#overview
    Convert MXNet models: https://github.com/apache/incubator-mxnet/tree/master/tools/coreml

- *AWS DeepLens*



- *Weight Pruning*

  The TensorFlow Model Optimization Toolkit *is a suite of tools for optimizing ML models for deployment and execution. The toolkit supports techniques used to:*

  - ✓ *Reduce latency and inference cost for cloud and edge devices*
  - ✓ *Deploy models to edge devices with restrictions on processing, memory, power-consumption, network usage, and model storage space.*
  - ✓ *Enable execution on and optimize for existing hardware or new special purpose accelerators.*

  The area of model optimization can involve various techniques:

  - ✓ *Reduce parameter count with pruning and structured pruning.*
  - ✓ *Reduce representational precision with post-training quantization.*

  Steps to do Weight Prunings:
  https://colab.research.google.com/drive/1LizRdQjDRMUxdFFc3zGR10Xq-8mTHA-2#scrollTo=6YHbYA-mZvhU

- *TensorFlow Lite*

  TensorFlow Lite is an example format you can use to deploy to mobile devices.
  The workflow for using TensorFlow Lite:

  1₩ Pick a model
  Bring own model, find a model online, or pick a pre-trained model to drop in or retain
  2₩ Optimize model
  Use Model Optimization Toolkit to reduce model's size and increase its efficiency with minimal impact on accuracy

3₩ Convert the model

Use TFLiteConverter to convert custom model to TensorFlow Lite format. To convert to a TensorFlow Lite graph, we need to use the TFLiteConverter.

```
[ ] tflite_model_file = './tf_opt/sparse_mnist.tflite'
    converter = tf.lite.TFLiteConverter.from_keras_model_file(pruned_keras_file)
    tflite_model = converter.convert()
    with open(tflite_model_file, 'wb') as f:
      f.write(tflite_model)
```

4₩Deploy to Device

Run model on-device with TensorFlow Lite Interpreter, with APIs in many languages

- *Hyperparameter Tuning*

  Basically, it's a process of choosing the right parameters for helping impact accuracy, model metrics and so on.

  Examples:

  https://colab.research.google.com/drive/1jAhm-QlOH7AComfebmq9A1k27ZIRFt5M

  https://colab.research.google.com/drive/1C0Wn9oAR4KrXn8Jp3twrD_wPIGK3uK7v#scrollTo=fLb3tgHUbkL1

  1/Grid search:

  try every combination of a present list of values of the hyper-parameters and evaluate the model for each combination. With the number increases, the evaluation numbers increase exponentially.

  2/Random search:

  Use random combination to find the best solution, and it's more efficient.

- *NAS*

  1/Search Space:

  The search space defines which architectures can be represented in principle. Like Conv layers, Max pooling, etc.

  2/Search Strategy:

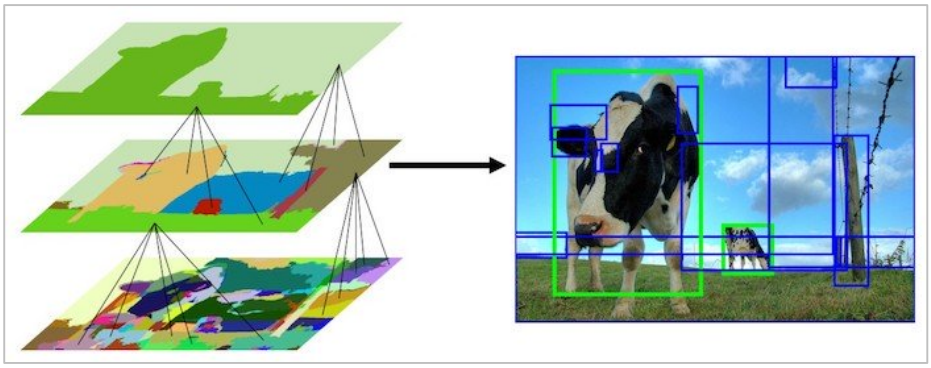
  The search strategy details how to explore the search space. Like Reinforcement Learning; neuro-evolutionary approaches, etc.

  3/Performance Estimation Strategy:

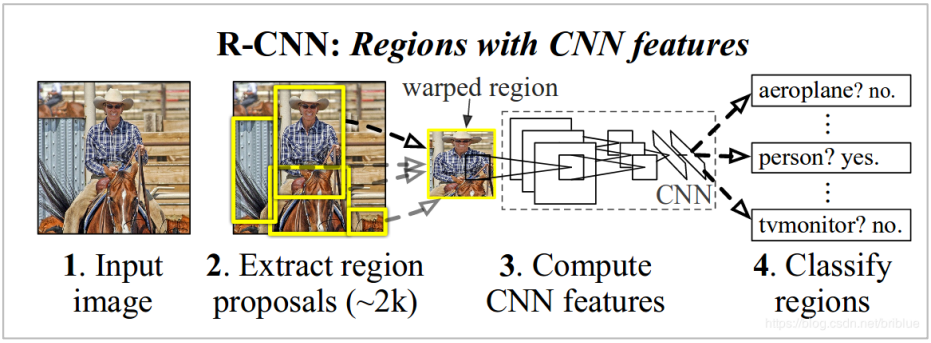
  The objective of NAS is typically to find architectures that achieve high predictive performance on unseen data.

Week 4:

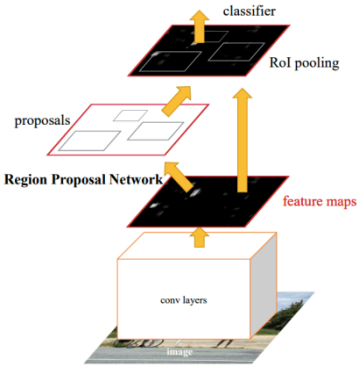
- *Object Detection*
  - *Data Requirements:*
    2 coordinates, the class label,100+ images per class, images covering different scenarios
  - *Process: Region Proposals → Feature Extraction → Evaluation Metric.*
    Firstly, use 'selective search' algorithm to generate object proposals, then use CNN to extract features.
  - *Selective Search: a clustering-based approach which attempts to group pixels and generate proposals based on the clusters.*



  - *Evaluation Metric： IoU& mAP*
    IoU: 'intersection over union'
    mAP: 'mean average precision'
  - *Precision & Recall:*
    https://www.zhihu.com/question/19645541/answer/91694636
    - *True Positive, True Negative, False Positive, False Negative*
  - *Region Proposal Algorithms:*
    - *objectness*
    - *selective search*
    - *category-independen object proposals*
    - *constrained parametric min-cuts(CPMC)*
    - *multi-scale combinatorial grouping*
    - *Ciresan*
- *R - CNN*
  - *Selective Search + CNN + SVM*



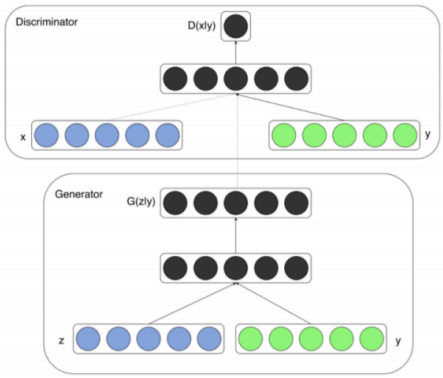
- *Faster R – CNN*



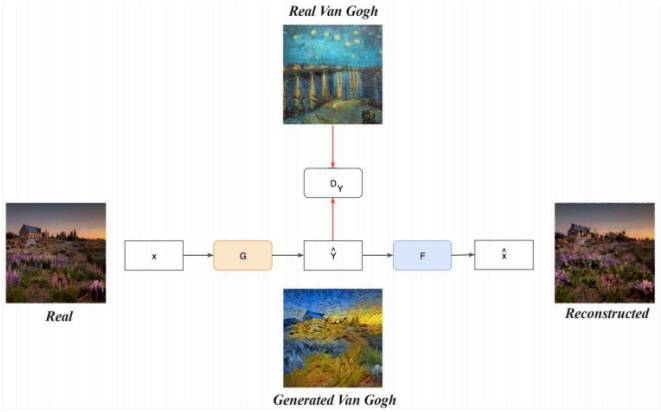
- *Data Scramping – fake-useragent*
  - *BeautifulSoap*
  - *UserAgent*

Week 3:

- *cGAN*
  - *involve the conditional generation of images and can be conditional on a class label (the MNIST dataset has class labels of the corresponding integers)*
  - *To incorporate the class labels into the discriminator and generator models*
    - *use an embedding layer followed by a fully connected layer with a linear activation*
    - *concatenate it with the image features as an additional channel or feature map*



- *cycleGAN*
  - *A generator G to convert a real image to a Van Gogh style picture.*
  - *A generator F to convert a Van Gogh style picture to a real image.*
  - *A discriminator D to identify real or generated Van Gogh pictures*

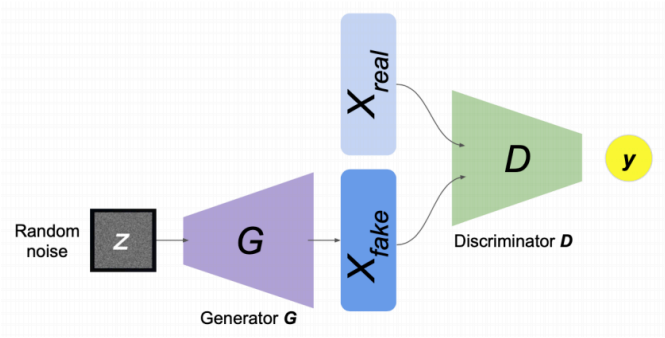


Week 2:

- *Generative Modeling*
  - *Goal: Take training samples as input from some distribution and learn a model that represents that distribution*
- *AutoEncoder*
  - *An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal "noise". It's an unsupervised approach for learning a feature representation from unlabeled training data. With the encoder (CNN/dense layer), we extract the features of meaningful factors of variation in data.*
- *VAE*
  - *Variational autoencoders (VAEs) are a deep learning technique for learning latent representations. They have also been used to draw*

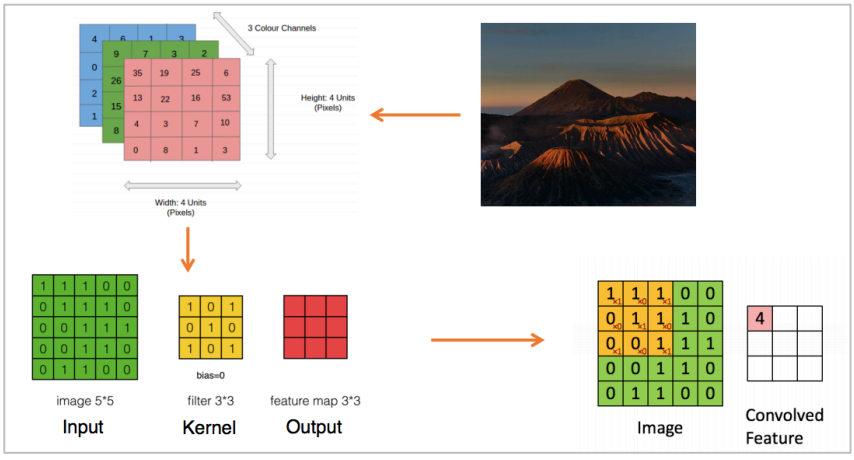
images, achieve state-of-the-art results in semi-supervised learning, as well as interpolate between sentences.

- *GAN*
  - *GANs are a way to make a generative model by having two neural networks compete with each other.*

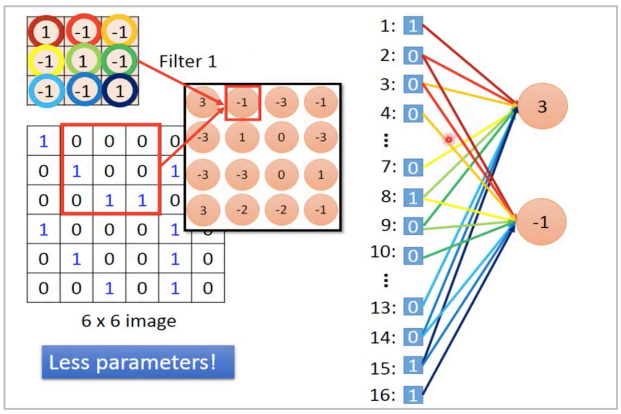


Week 1:

- *CNN – LeNet5 – AlexNet – DNN – Transfer Learning*
  - *CNN:*
    - *The whole process:*
      Convolution – Max Pooling – Flatten – Fully Connected Network – Output. Convolution and Max Pooling can repeat many times. Convolution is for dealing with patterns that are much smaller and appear in different regions of an image. Max Pooling is for dealing with images that have been subsampled the pixels
    - *Feature Map*

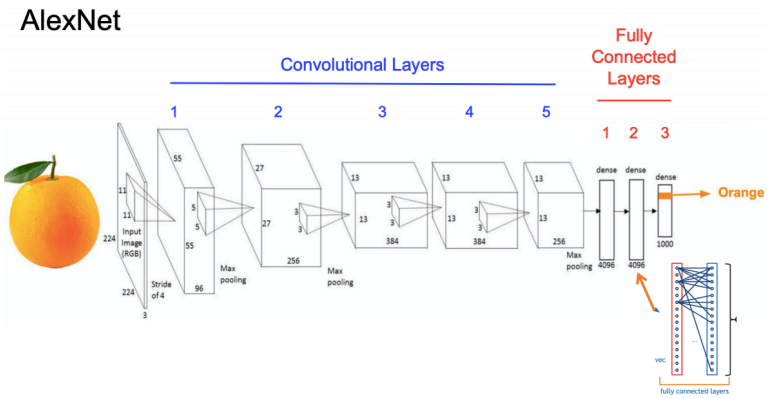


    - *Convolution v.s. Fully Connected*



- *Max Pooling:*
  Find the most important part of the feature map, help over-fitting, and reduce the computational cost.
- *CNN in Keras:*
  https://colab.research.google.com/drive/18QRkL-zyQ8XDl_UjtR10vq4Ve45Q8TvQ
- *Activation Function:*
  Sigmoid, tanh, ReLU, softmax

- *AlexNet:*
  - *A Deep Convolutional Neural Networks.*
  - *Pooling, ReLU activation function for nonlinearity.*
  - *Dropout was implemented to reduce overfitting*
  - *Data Augmentation technique was used.*
  - *Stochastic Gradient Descent was used as optimizer with momentum.*



- *Transfer Learning:*
  is a machine learning technique where a model trained on one task is re-purposed on a second related task.

- *CPU, GPU, FPGA, TPU*
    - *CPU: aka Central Processing Unit, is designed to solve multiple problems but not parallel.*
    - *GPU: aka Graphics Processing Unit, is good at processing the same set of operations in parallel. (Single instructions, multiple data)*
    - *FPGA: aka Field Programmable Gate Array, is good at processing the same of different operations in parallel. (Multiple instructions, multiple data)*
    - *TPU: aka Tensor Processing Unit, is specially for neural network machine learning.*
- *TensorFlow Russian Dolls*
    - *TensorFlow – Keras – Ludwig – GCP AutoML – GCP Vision API*
- *Biological Vision's development*
    - *Trilobite developed vision*
    - *Torsten Wiesel and David Hubel carried out a research on how visual information is processed in mammalian cerebral cortex. They used a light projector to present slides to anesthetized cats while they recorded the activity of neurons in the cats' primary visual cortex.*
    - *Facts about cerebral cortex: 1/ one of the most recent evolutionary developments of the brain. 2/ although cortex tissue is grey, the bulk of the brain is white matter. 3/ the white matter is for carrying info over longer distances than the grey matter and the grey matter is the part for the most complex computations.*
    - *Hubel and Wiesel began to present simple shapes like dot to the cats, but it didn't work. Then they serendipitously discovered that the neurons that receive visual input from the eye are in general most responsive to simple, straight edges.*
    - *Illustrate how, via hierarchically organized layers of neurons feeding information into increasingly higher-order neurons, gradually more complex visual stimuli can be represented by the brain.*
    - *Regions of the visual cortex*

- *Machine Vision's development*
    - *BV serves as the inspiration for the modern deep learning approaches to machine vision.*
    - *Kunihiko Fukushima proposed an analogous architecture for machine vision, which he named the neocognitron.*

    - *LeNet-5 by Yann LeCun and Yoshua Bengio, and in it had an algorithm that could correctly predict the handwritten digits that had been drawn without needing to include any expertise about handwritten digits in code.*
    - *LeNet-5 shows the difference between deep learning and traditional machine learning. The traditional is more focusing on feature engineering, while deep learning spent the majority of time on the design and tuning of model architectures.*
    - *MNIST*
    - *ImageNet and the ILSVRC*
    - *AlexNet*

- *Colab & Jupyter Notebook*
    - *Colab Interface:*
        - *gpu₩tpu*
        - *table of contents*
        - *code snippets*
        - *files (upload)*
        - *Forms menu in colab*
        - *Upload to colab (for later processing)*
        - *Github integration*
        - *Save to github and gist*
    - *Manage Colab documents:*
        - *Mount gdrive workflow*
        - *Wire up kaggle*
        - *Runtime*
        - *Universal image and data*