SIEMENS

Multiple Objects Detection in Single Image by Deep Learning

Presenter: Zhenyu Wu
Instructor: Neeraj Panhalkar, Pengcheng Liu & Michael Nguyen

Restricted © Siemens AG 2016. All rights reserved

Outline

SIEMENS

- Introduction
  - Project Description
  - Intro to Deep Learning
  - Intro to Convolutional Neural Network
  - How to Train a ConvNet from Scratch
  - Intro to TensorFlow
- Classification
  - Related work
  - Data set
  - Architecture
  - Performance
  - Transfer learning
- Localization
  - Related work
  - Data set
  - Performance
- Conclusion
- Future work
- References

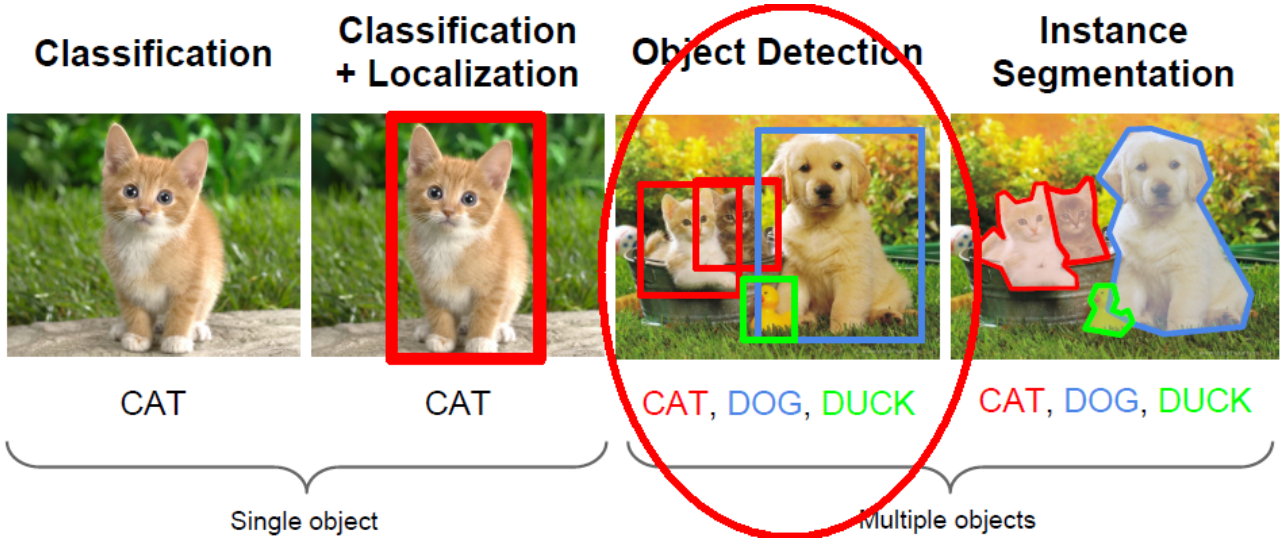
SIEMENS

Introduction

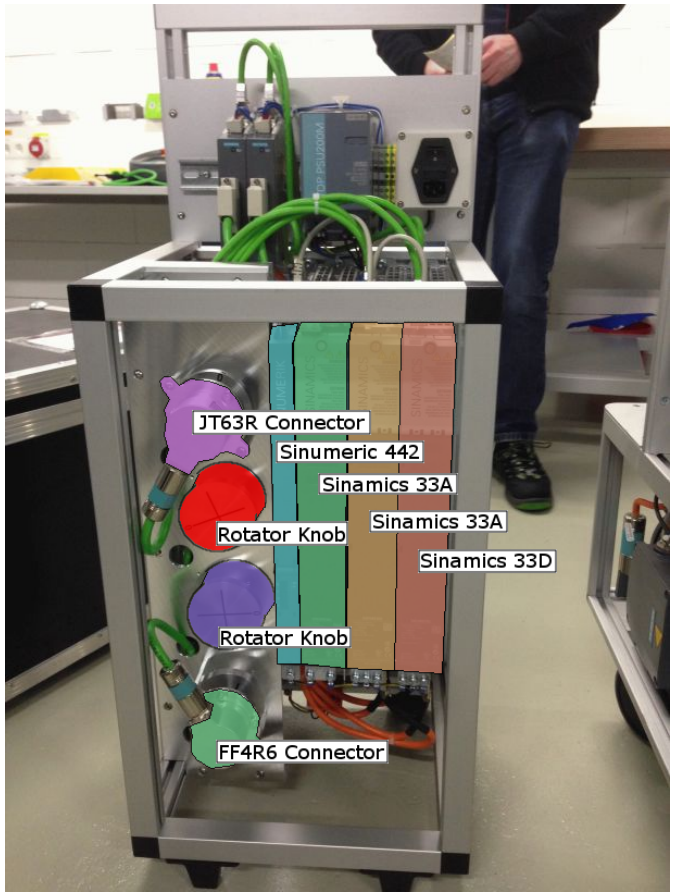
Restricted © Siemens AG 2016. All rights reserved

Project Description

- Given an image with user's device
  - Many mechanic parts of different types are contained in the image
- Need to figure out
  - The number of the objects of interest
  - The location of the objects of interest
  - The category of the objects of interest
- Predicted difficulty
  - Objects Occlusion
- Classification < Localization < Detection (Subset)



Computer Vision Tasks[1]

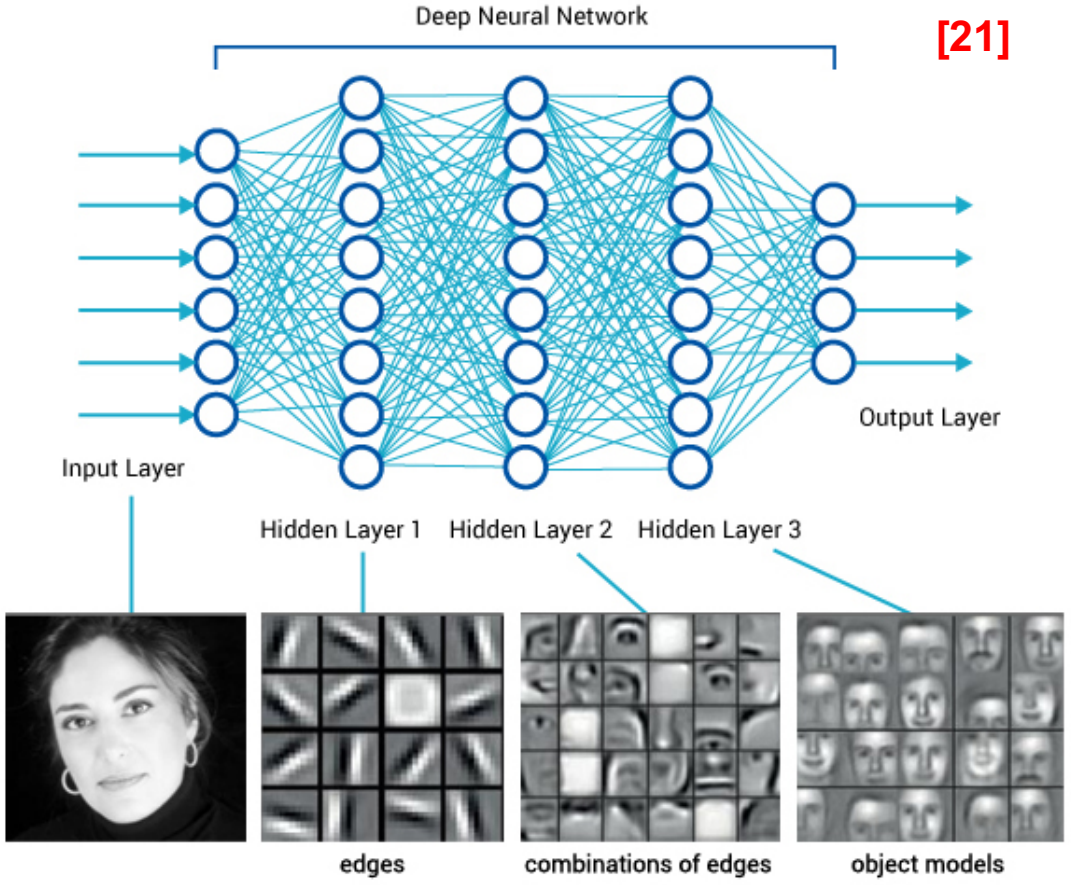


User Case

Intro to Deep Learning

SIEMENS

- A subset of machine learning
- Deep neural network is a special artificial neural network with at least two hidden layers
- Universal Approximation Theorem
- Deep network has better expressive power than shallow network
  - The deeper the layer is, the higher level feature it is able to extract
  - Intuition: composition of functions



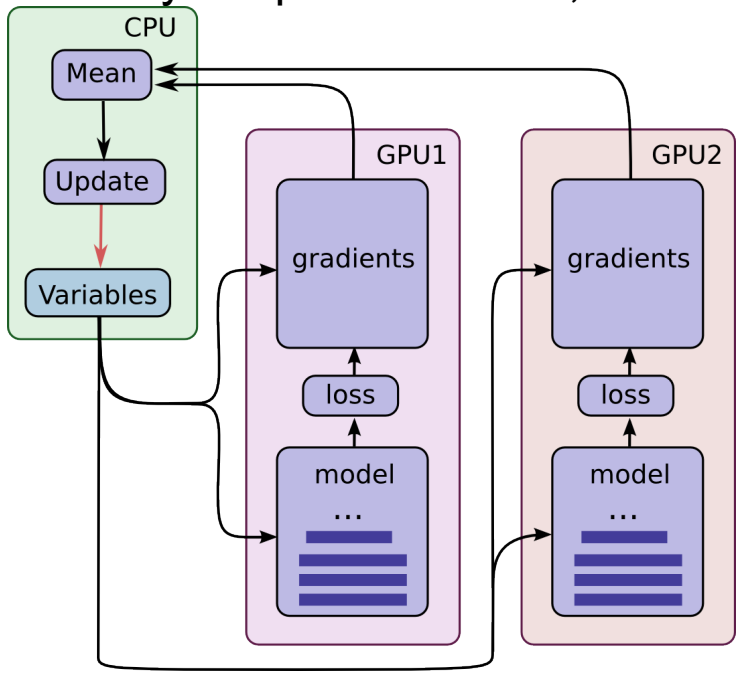
Intro to Convolutional Neural Network (ConvNet) [3]

SIEMENS

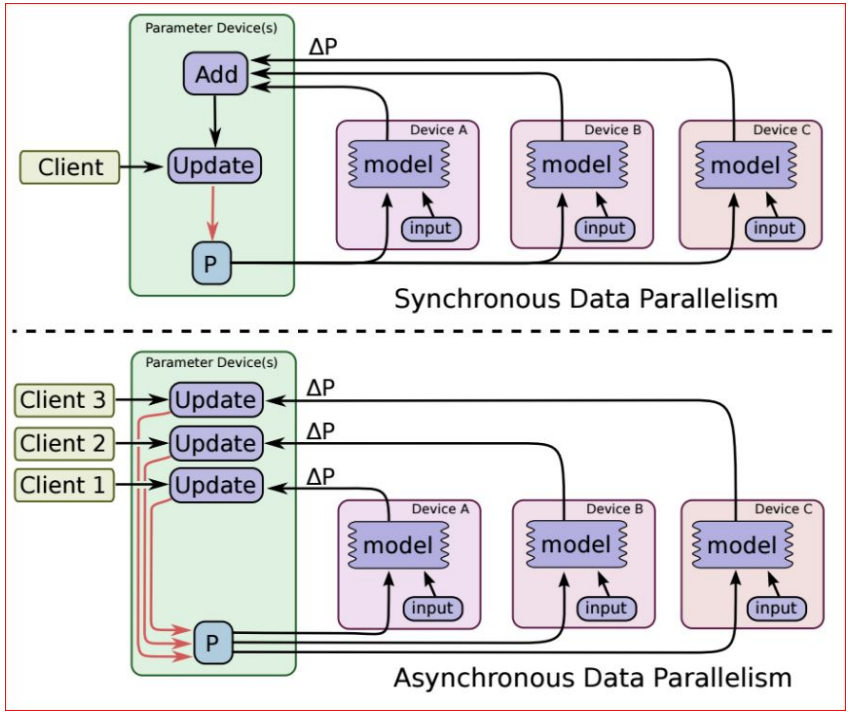
- Achieve great success in the field of Computer Vision
- Trained end-to-end, from raw pixels to ultimate categories
- Convolution + Pooling: fully utilize spatial connection in the image
- Sparse Connectivity + Shared Weights: greatly reduce number of parameters
- ConvNet is a stack of CONV, POOL, FC layers
- Trend towards **smaller** filters, **deeper** architectures and **remove** POOL/FC layers (refer to the evolution of Google's Inception module)
- Dropout trick + ReLU (rectified linear unit) activation [4]

How to Train a ConvNet from Scratch [9][19]

- Training a ConvNet is an optimization on an extremely high dimension non-convex loss function
- Training becomes easier thanks to GPU and Distributed Computing
- Two ways of parallelization, **Data Parallel** & Model Parallel



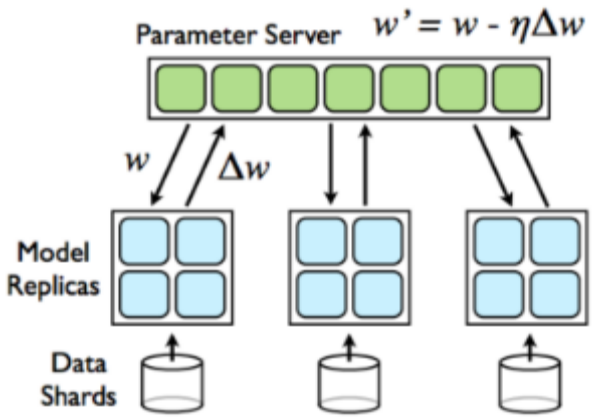
Single machine with multiple GPUs



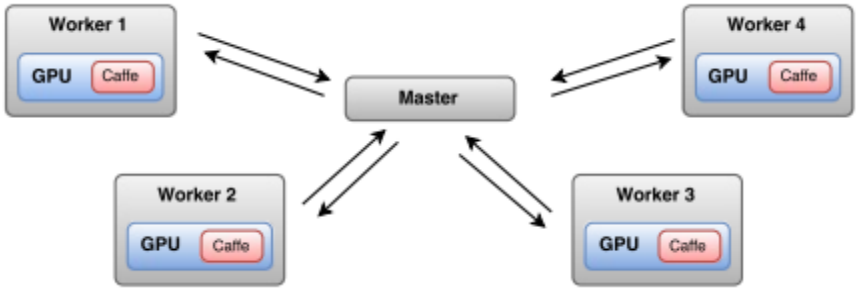
Multiple machines in distributed setting

How to Train a ConvNet from Scratch (cont.)

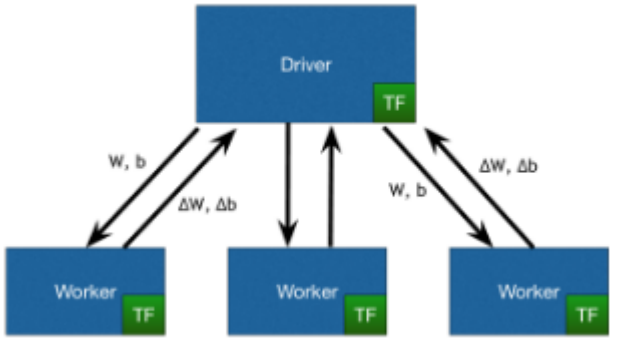
- Training on single machine with multiple GPUs
  - Very well supported in TensorFlow
- Distributed computing framework for TensorFlow
  - TensorSpark [8]
  - SparkNet (Claim to support TensorFlow in the future)
  - Spark + YRAN computation cluster on AWS



Google's DistBelief Architecture



SparkNet Architecture



TensorSpark Architecture

How to Train a ConvNet from Scratch (cont.)

SIEMENS

- Amazon Web Service
  - Spark + YARN

```
with tf.device("/job:ps/task:1"):
  weights_2 = tf.Variable(...)
  biases_2 = tf.Variable(...)

with tf.device("/job:worker/task:7"):
  input, labels = ...
  layer_1 = tf.nn.relu(tf.matmul(input, weights_1) + biases_1)
  logits = tf.nn.relu(tf.matmul(layer_1, weights_2) + biases_2)
  # ...
  train_op = ...

with tf.Session("grpc://worker7.example.com:2222") as sess:
```

Software configuration

Vendor (●) Amazon ( ) MapR

Release emr-4.7.1

Applications

( ) Core Hadoop: Hadoop 2.7.2 with Ganglia 3.7.2, Hive 1.0.0, Hue 3.7.1, Mahout 0.12.0, and Pig 0.14.0

( ) HBase: HBase 1.2.1 with Ganglia 3.7.2, Hadoop 2.7.2, Hive 1.0.0, Hue 3.7.1, Phoenix 4.7.0, and ZooKeeper 3.4.8

( ) Presto-Sandbox: Presto 0.147 with Hadoop 2.7.2 HDFS and Hive 1.0.0 Metastore

(●) Spark: Spark 1.6.1 on Hadoop 2.7.2 YARN with Ganglia 3.7.2

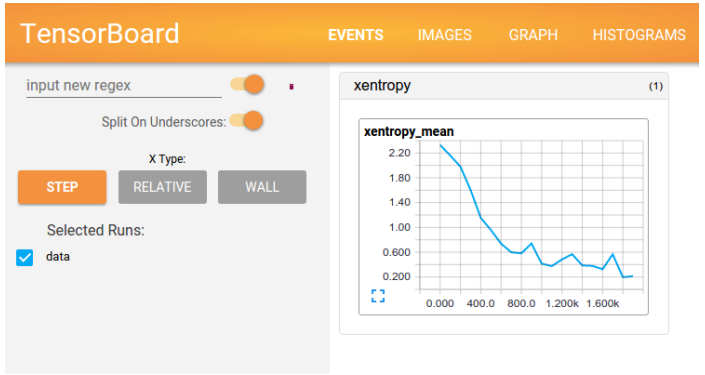
Hardware configuration

Instance type g2.2xlarge

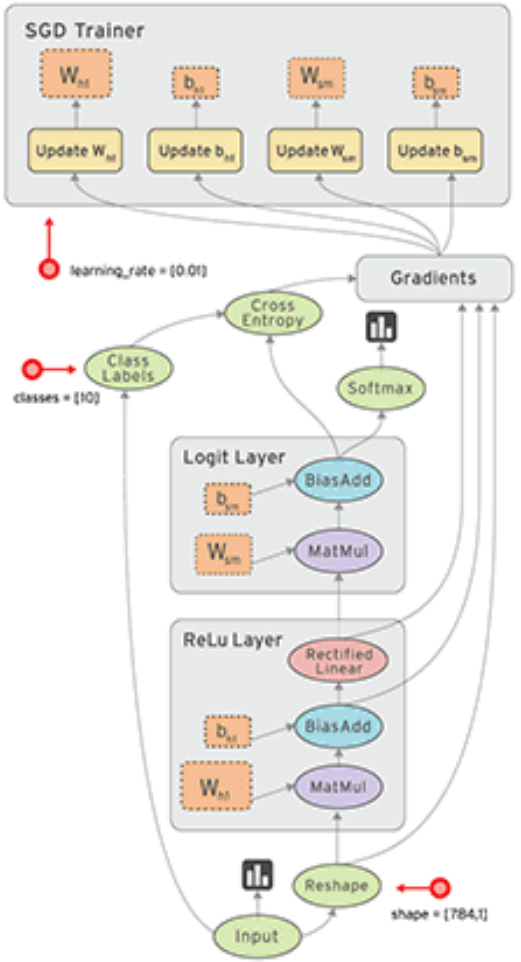
Number of instances 3 (1 master and 2 core nodes)

Intro to TensorFlow [16]

- Google's second-generation system for the implementation and deployment of large-scale machine learning models
- Support hardware platforms ranging from mobile device to single machine with GPUs to hundreds of machines
- Describe computation using a dataflow-like model (similar to Theano)
- Tensors (N-dimensional array) flow in a computation graph, where operations are represented as nodes and reliance between operations are represented as edges
- TensorBoad is a suite of visualization tools able to visualize graph and plot quantitative metrics about the execution of the graph



TensorBoard



Data-flow graph

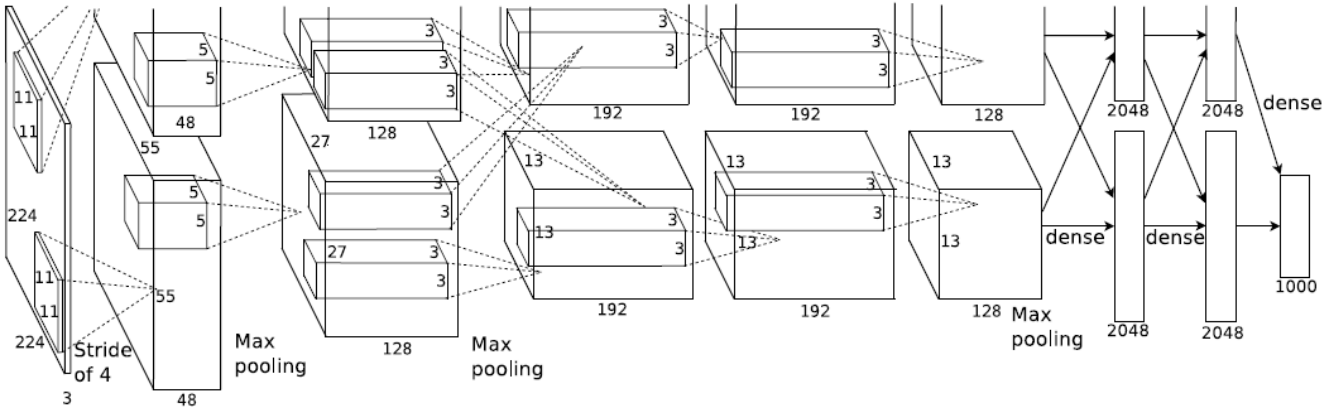
SIEMENS

Classification

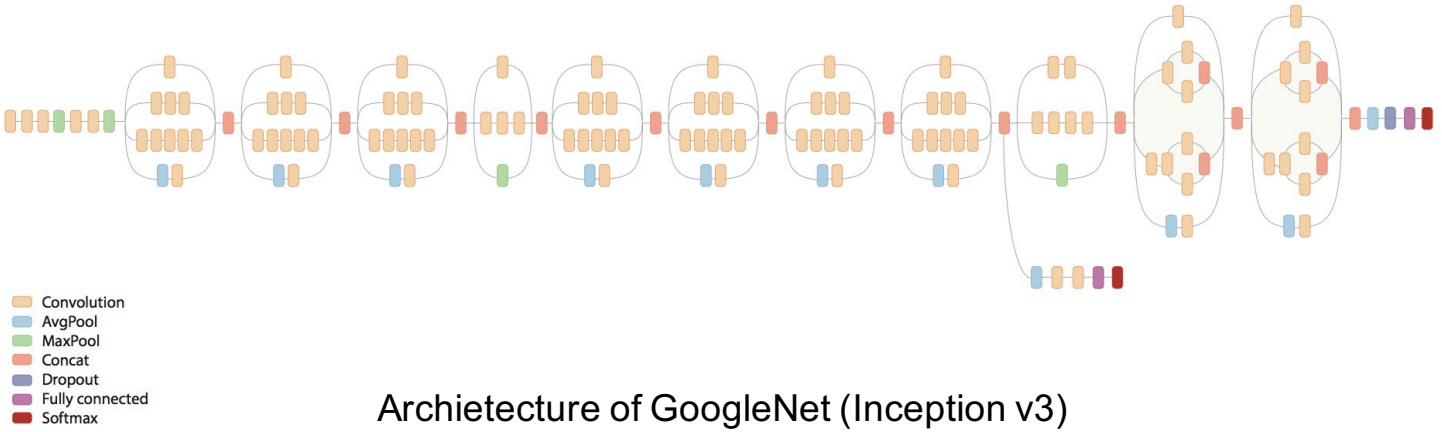
Restricted © Siemens AG 2016. All rights reserved

Related Work

- AlexNet (2012) [4]
  - 8 layers
  - 40.7% top-1 & 18.2% top-5
  - 60 million parameters
- VGGNet (2014) [5]
  - 19 layers
  - 24.4% top-1 & 7.1% top-5
  - 180 million parameters
- GoogleNet (Inception-v3) (2015) [6]
  - 48 layers
  - 18.77% top-1 & 4.2% top-5
  - 5 billion multiply-adds / inference
  - 25 million parameters



Archietecture of AlexNet

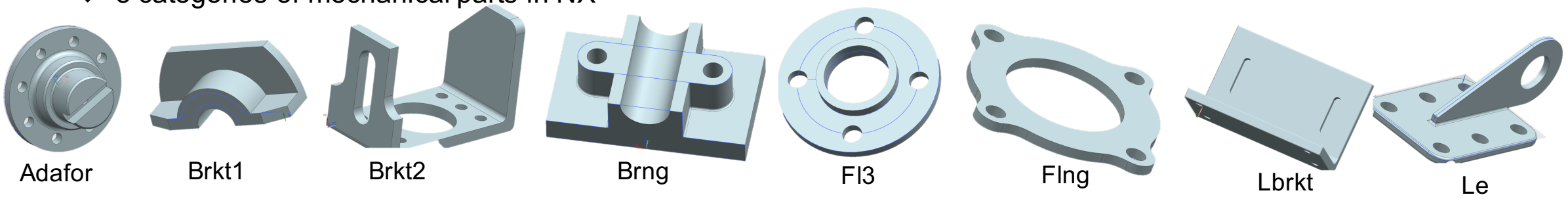


Archietecture of GoogleNet (Inception v3)

Data Set

SIEMENS

- 8 categories of mechanical parts in NX



- Expand the size of data set using NX journal
  - Rotate the part by some degree along 3 axis
  - Assign colors to features randomly
  - Add noisy background randomly
- Expand the size of data set using TensorFlow (Only do it in training)
  - Do distortion on each image
  - Flip left and right
  - Adjust the brightness and contrast by a random factor

Architecture

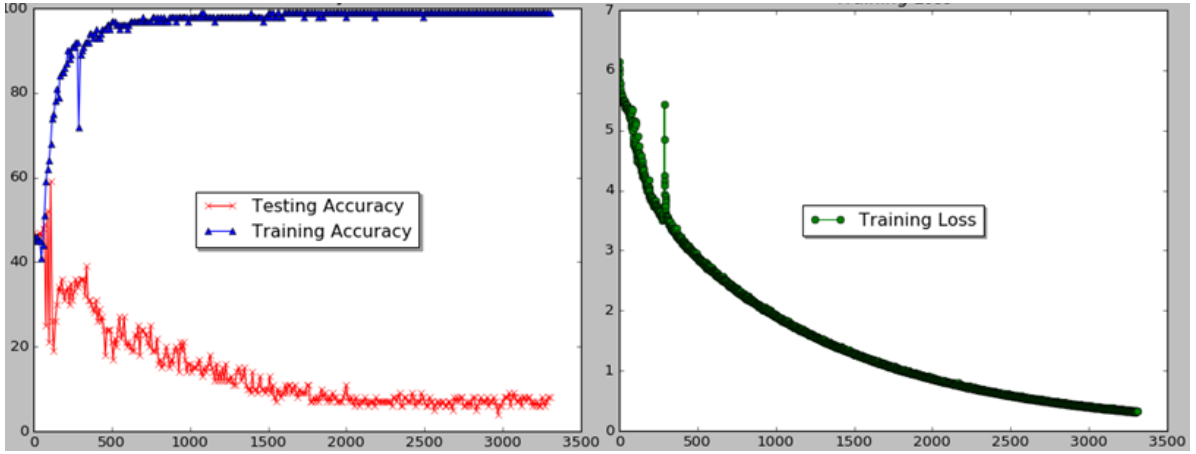
SIEMENS

Layer	Stage	Filter size	Conv stride	Pooling size	Pooling stride	Tensor Shape
1 (input)	\	\	\	\	\	(256, 128, 128, 3)
2	Conv+Max	[5,5,3,64]	[1, 1, 1, 1]	[1, 3, 3, 1]	[1, 2, 2, 1]	(256, 64, 64, 64)
3	Conv+Max	[5,5,64,64]	[1, 1, 1, 1]	[1, 2, 2, 1]	[1, 2, 2, 1]	(256, 32, 32, 64)
4	FCC	\	\	\	\	(256, 384)
5	FCC	\	\	\	\	(256, 192)
6 (output)	FCC	\	\	\	\	(256, 5)

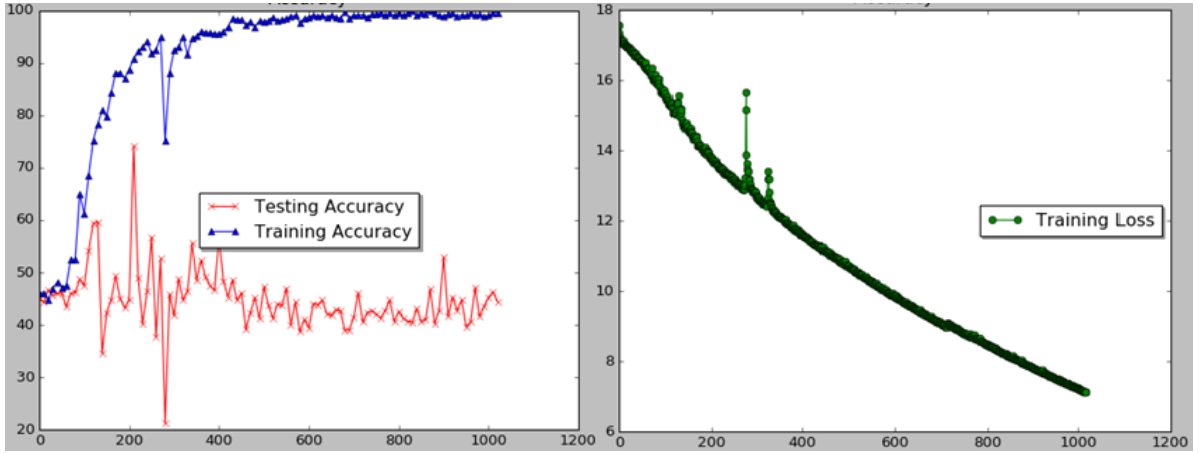
Performance (Using Imbalanced Data)

SIEMENS

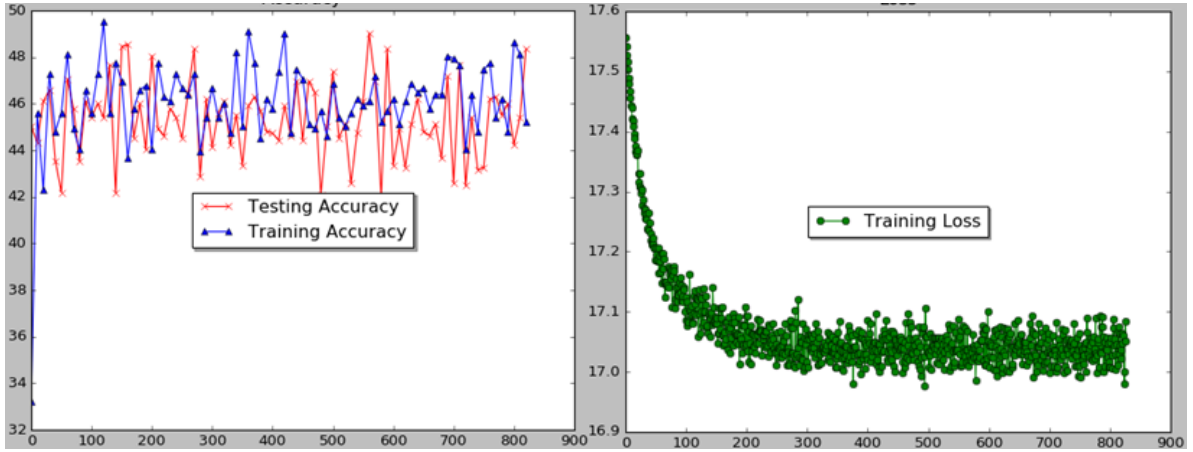
8 categories, 32x32, lr = 0.1, batch size = 1024, w/ dropout, LRN, distortion



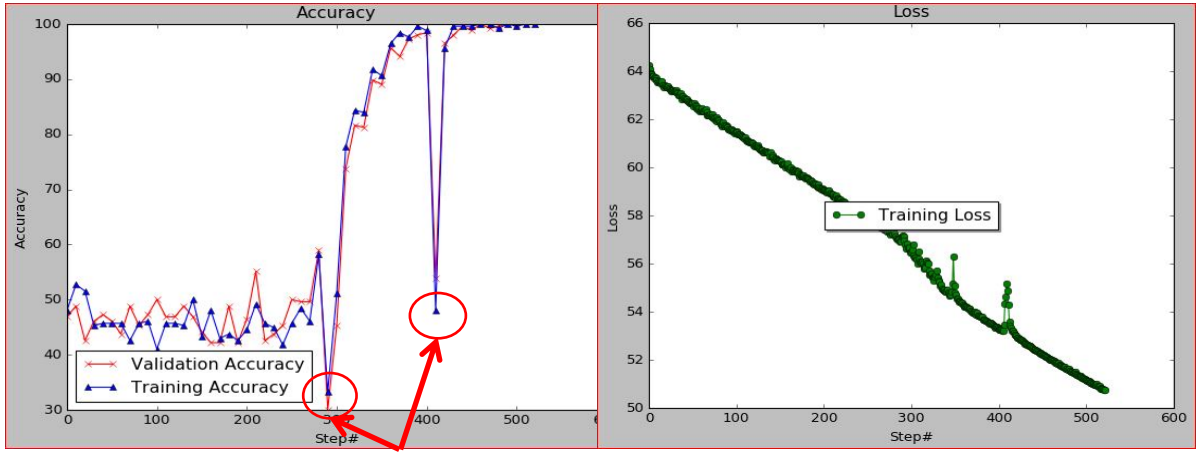
6 categories, 64x64, lr = 0.1, batch size = 512, w/ dropout, LRN, distortion



6 categories, 64x64, lr = 0.005, batch size = 512, w/ dropout, LRN, distortion



6 categories, 128x128, lr = 0.05, batch size = 256, w/o dropout, w/o LRN, w/o distortion



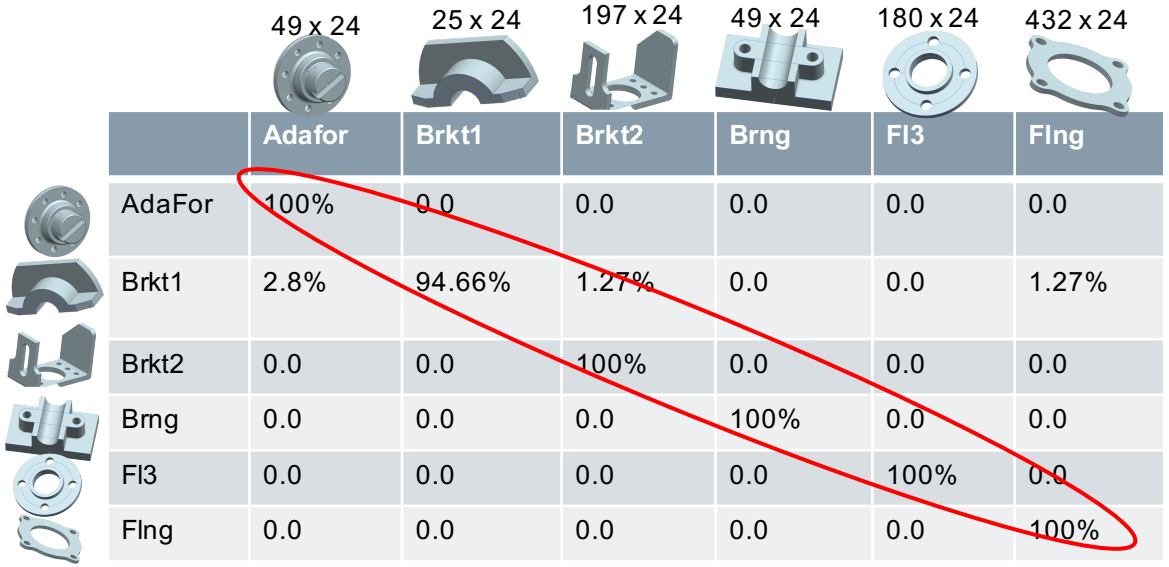
Neural Network reached saturation state?

Performance (cont.)

	Adafor	Brkt1	Brkt2	Brng	Fl3	Flng	Lbrkt	Le
AdaFor	5.52%	0.0	0.0	0.0	0.0	0.0	94.48%	0.0
Brkt1	0.0	1.46%	0.0	0.0	0.0	0.0	98.54%	0.0
Brkt2	0.0	0.0	2.82%	0.0	0.0	0.0	97.18%	0.0
Brng	0.0	0.0	7.71%	19.99%	0.99%	0.0	71.3%	0.0
Fl3	0.0	0.0	0.94%	0.0	25.42%	0.0	73.64%	0.0
Flng	0.0	0.0	0.0	0.0	0.0	0.48%	91.05%	8.47%
Lbrkt	0.0	0.0	0.0	0.0	0.0	0.0	100%	0.0
Le	0.0	0.0	0.0	0.0	0.0	0.0	77.8%	22.2%

	Adafor	Brkt1	Brkt2	Brng	Fl3	Flng
AdaFor	6.16% (100%)	0.0	72.73%	13.22%	1.765%	6.12%
Brkt1	0.0 (0.76%)	5.971% (95.63%)	82.21% (1.387%)	2.89% (0.9%)	0.0	8.93% (1.32%)
Brkt2	0.0	0.0	100% (99.96%)	0.0 (0.009%)	0.0 (0.027%)	0.0
Brng	0.0	0.0	55.31%	26.93% (100%)	0.83%	16.92%
Fl3	0.0	0.0	41.06%	0.0	58.94% (100%)	0.0
Flng	0.0	0.0	84.36%	0.0	0.0	15.64% (100%)

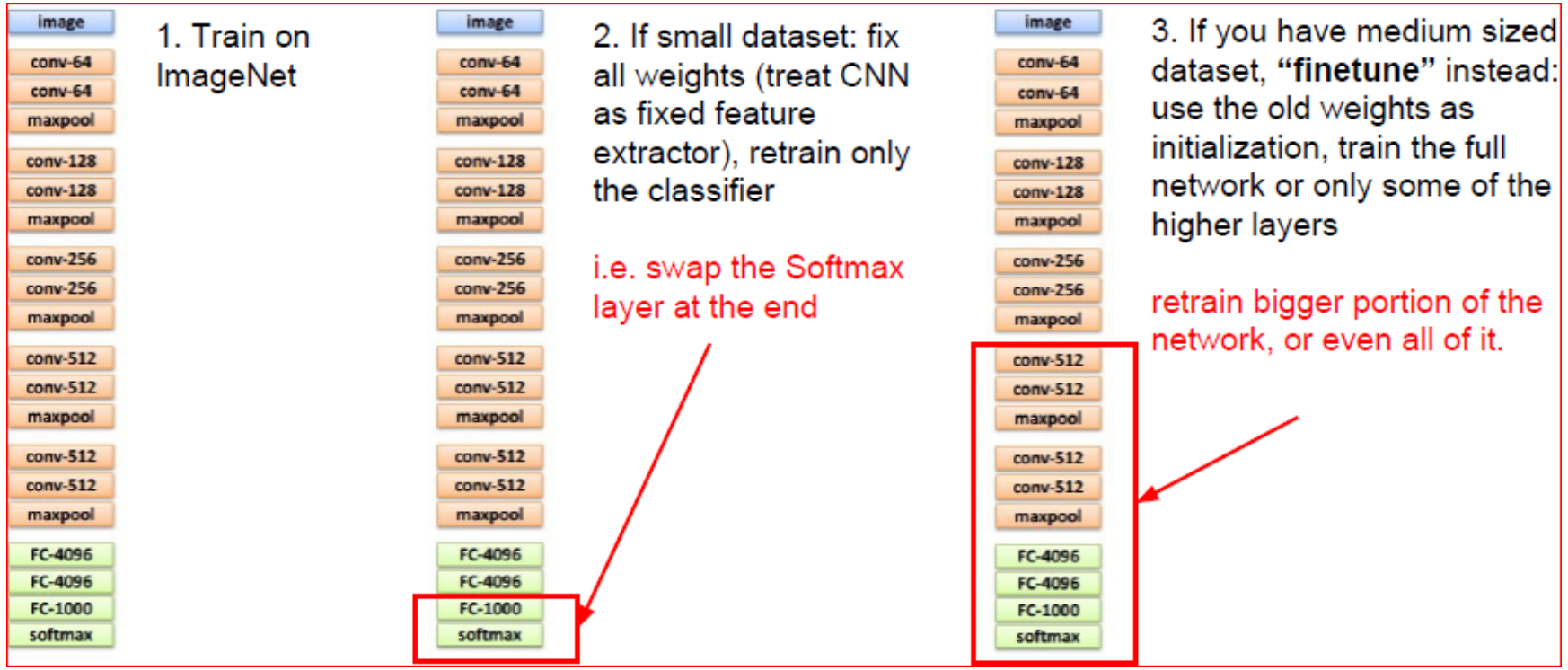
	Adafor	Brkt1	Brkt2	Brng	Fl3	Flng
AdaFor	0.0	0.0	0.0	0.0	0.0	100%
Brkt1	0.0	0.0	0.0	0.0	0.0	100%
Brkt2	0.0	0.0	0.0	0.0	0.0	100%
Brng	0.0	0.0	0.0	0.0	0.0	100%
Fl3	0.0	0.0	0.0	0.0	0.0	100%
Flng	0.0	0.0	0.0	0.0	0.0	100%



	Adafor	Brkt1	Brkt2	Brng	Fl3	Flng
AdaFor	100%	0.0	0.0	0.0	0.0	0.0
Brkt1	2.8%	94.66%	1.27%	0.0	0.0	1.27%
Brkt2	0.0	0.0	100%	0.0	0.0	0.0
Brng	0.0	0.0	0.0	100%	0.0	0.0
Fl3	0.0	0.0	0.0	0.0	100%	0.0
Flng	0.0	0.0	0.0	0.0	0.0	100%

Transfer Learning

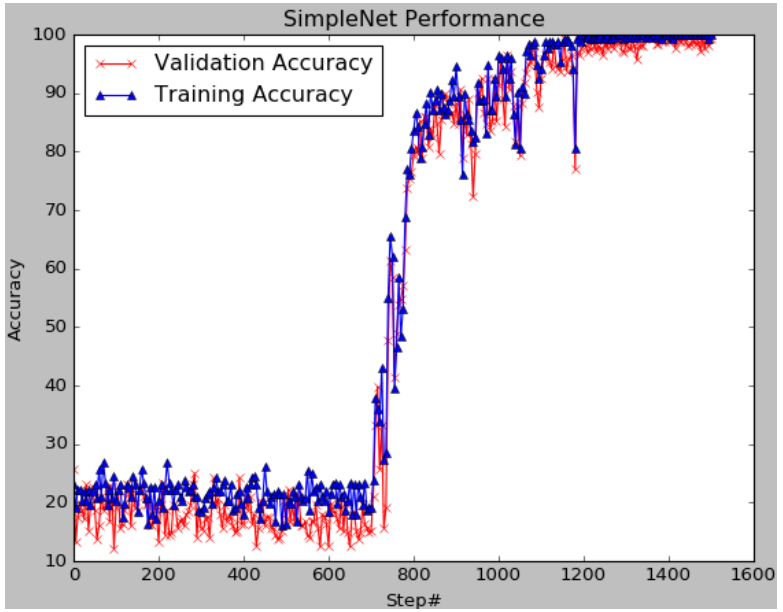
- Taking fully-trained model for a set of categories (like ImageNet)
- Retrain the final layer from scratch, while leaving all the others untouched
- Caffe Model Zoo: Lots of pretrained ConvNets (Caffe model is convertible to TensorFlow) [22]



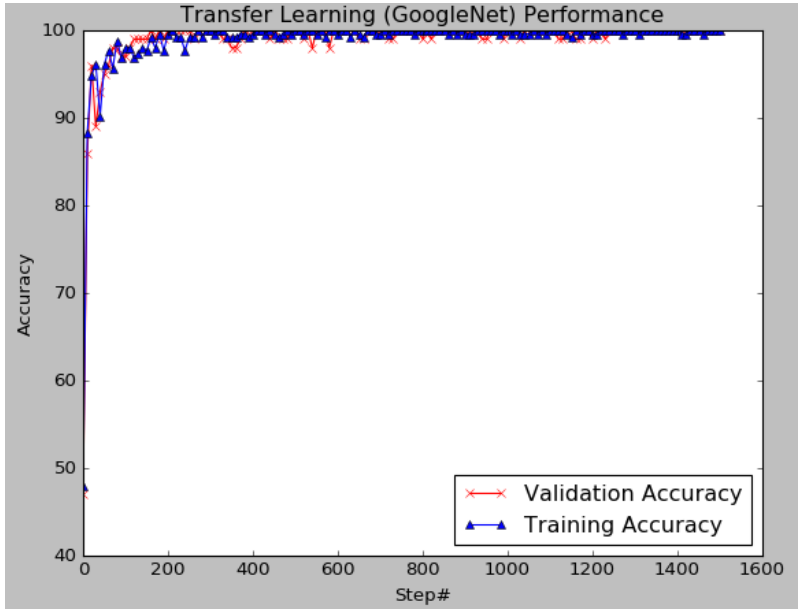
Transfer Learning (contd.)

SIEMENS

- Compare SimpleNet and Transfer Learning (GoogleNet)
  - Both use mini-batch SGD with same batch size but with different learning rate
  - Transfer Learning on GoogleNet is learning much faster
  - GoogleNet is very robust
  - Previous trained 47 layers are able to extract high-level features and the final output layer is simply the logic of how to combine the features to do prediction



Learning rate = 0.05



Learning rate = 0.01

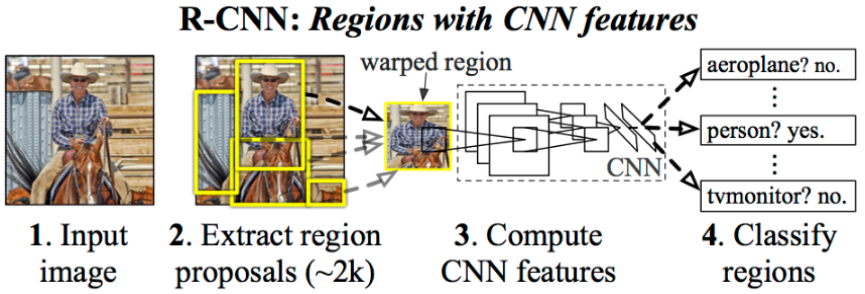
SIEMENS

Localization

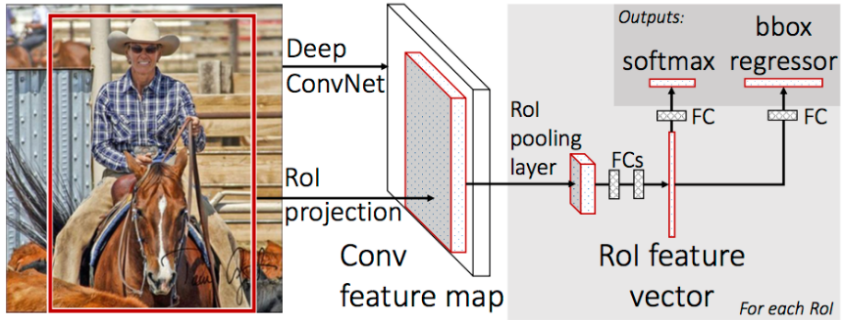
Restricted © Siemens AG 2016. All rights reserved

Related Work

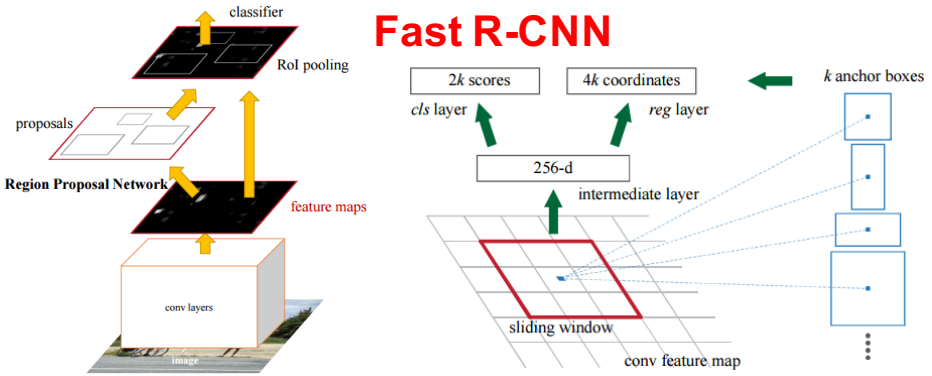
- R-CNN [10]
  - Milestone in using ConvNet for object detection
  - Selective Search to generate 2k region proposals
  - Use ConvNet to extract features for each region
  - Train SVM to do classification
  - Use regression to fine-tune the positions of bounding boxes
- Fast R-CNN [11]
  - Use ROI layer to pool region proposals
  - Use softmax for classification & a bbox regressor
- Faster R-CNN [12]
  - Use Region Proposal Network to generate proposals
  - Train RPN & R-CNN jointly



R-CNN



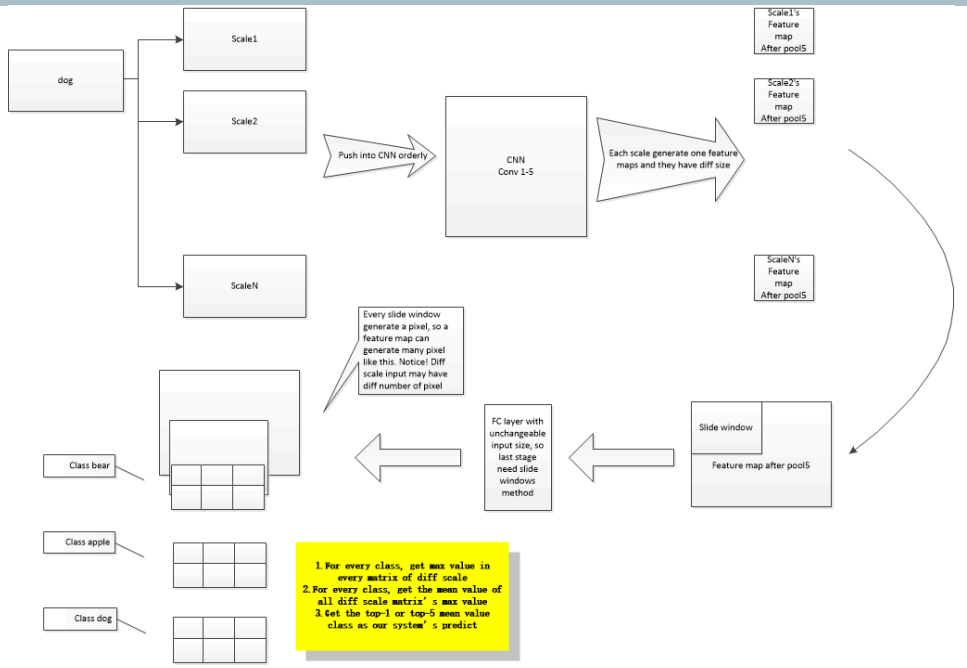
Fast R-CNN



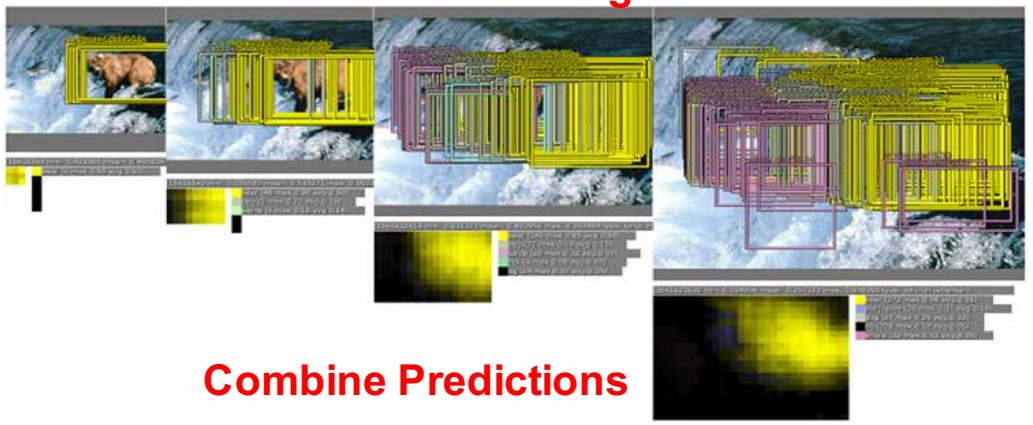
Faster R-CNN

Related Work (cont.)

- OverFeat [13]
  - Use sliding window detector
  - Propose a feature extractor named “overfeat”
  - Use multi-scale classification to support images with different resolution
  - Do regression and classification together
  - Classification score is the confidence of the bbox



Multi-Scale + Sliding Window



Combine Predictions

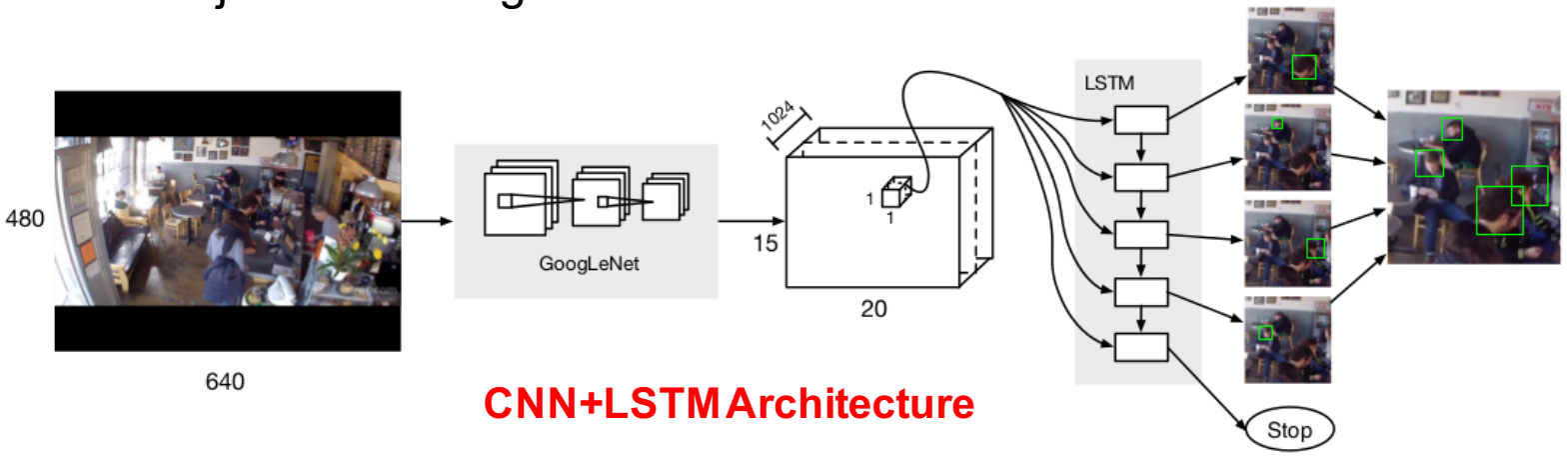
Related Work (cont.)

SIEMENS

- CNN + LSTM [14]
  - Use GoogleNet to generate a 15 x 20 x 1024 image feature map
  - For each cell in the feature map, use a LSTM controller to predict a set of bboxes
  - Use Hungarian Algorithm to assign an unique candidate bbox to ground truth box
  - Calculate loss function based on matching result
  - In the test process, use “stitching algorithm” to match new bboxes with existing bboxes, and destroy non-overlapping bboxes
  - LSTM can “memorize” history predictions thus avoiding multiple predictions on the same object
  - LSTM can predict arbitrarily number of object in an image

$$L(G,C,f) = \alpha \sum_{i=1}^{|G|} l_{pos}(\mathbf{b}_{pos}^{i}, \tilde{\mathbf{b}}_{pos}^{f(i)}) + \sum_{j=1}^{|C|} l_c(\tilde{\mathbf{b}}_c^{j}, y_j)$$

Loss Function

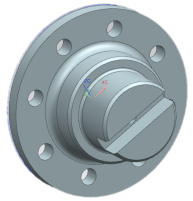


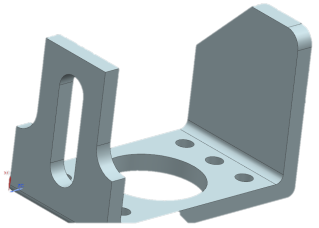
CNN+LSTM Architecture

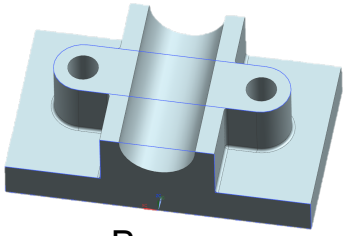
Data Set

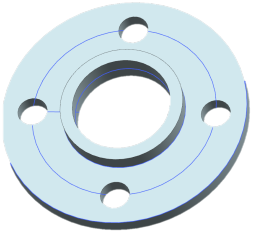
SIEMENS

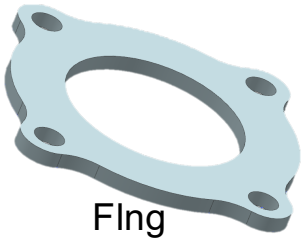
- Finally, we adopted the CNN + LSTM architecture in the previous silde
- 5 categories of mechanical in NX


Adafor

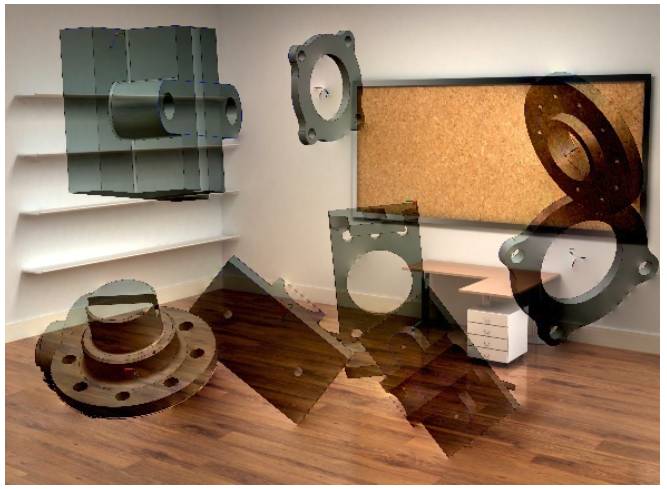

Brkt2


Brng


Fl3


Flng

- Manually compose an image with multiple objects occluded using OpenCV
  - Export mechanic parts as JPG files with white background
  - Extract objects out of background by convert image to grayscale
  - Seamless clone the objects into a noisy background
  - Record the location of bboxes (x, y, width, height)
  - Output images together with 'idl' file describing images location and bboxes coordinates (both for training and testing)

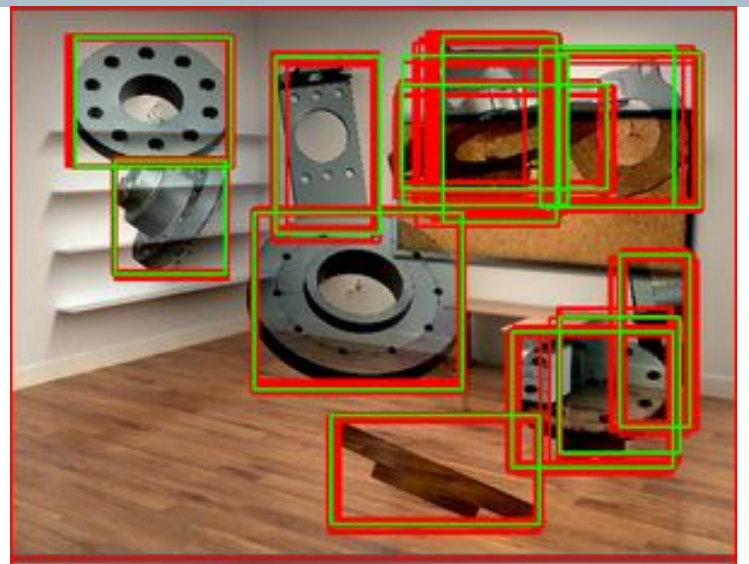


Manually Composed Image

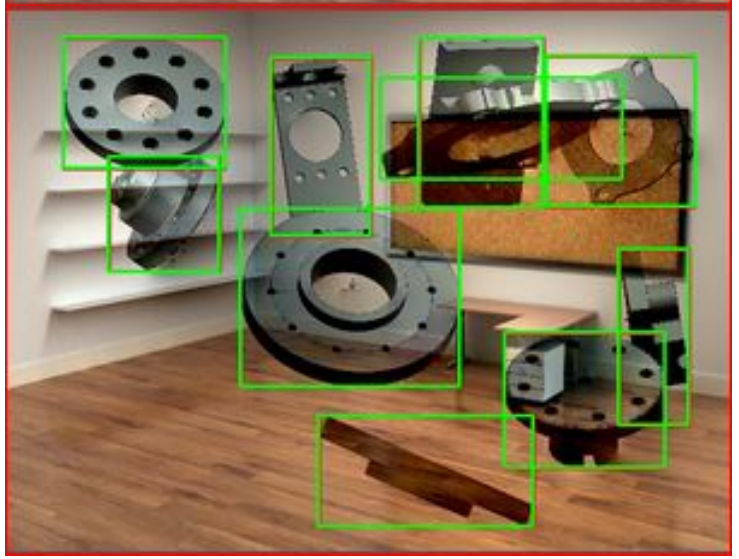
Performance

SIEMENS

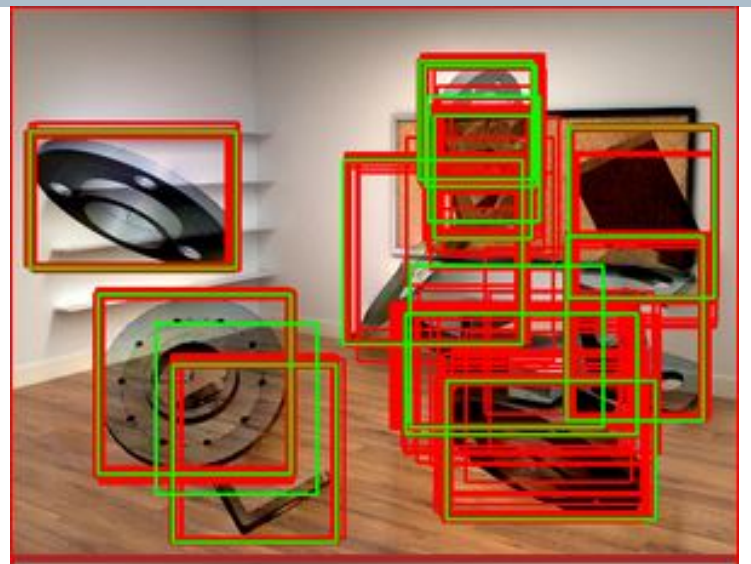
Predicted Output



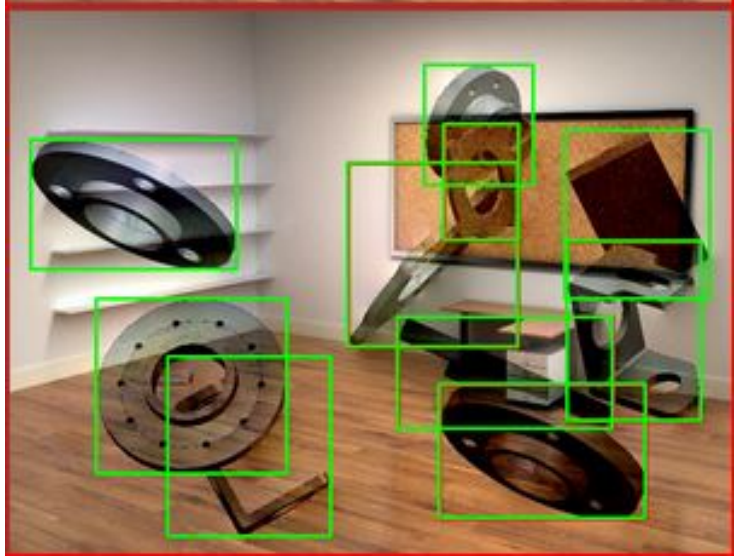
True Output



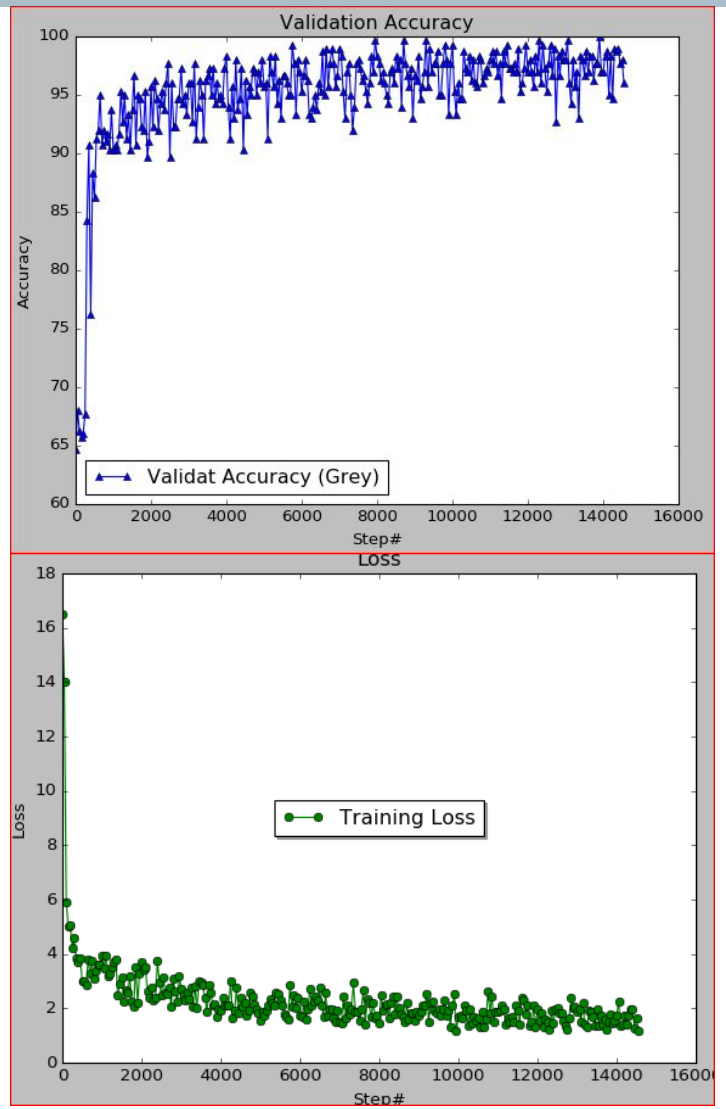
Predicted Output

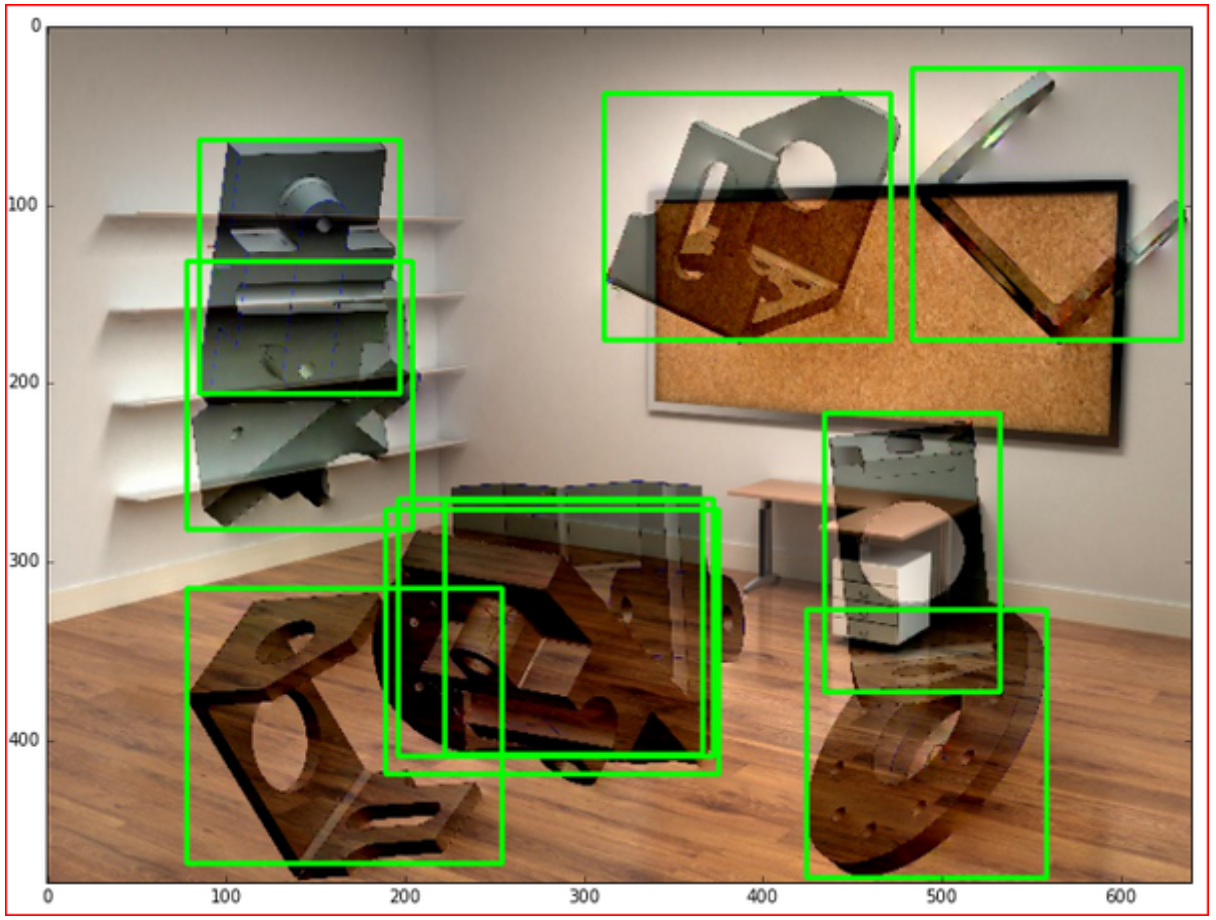


True Output



Performance (cont.)





SIEMENS

Conclusion

Restricted © Siemens AG 2016. All rights reserved

Conclusion

SIEMENS

- Deep Learning is being widely used in different fields
  - Computer Vision, Natural Language Processing, Speech Recognition…
  - Big companies like Google, Facebook, Amazon, Microsoft are devoting large amount of money into it…
- ConvNet is commonly adopted to solve Computer Vision problems [3]
  - Almost all of the state of arts in CV is using ConvNet
  - Theoretically, ConvNet is able to do classification, localization and detection together (refer to Overfeat & R-CNN)
- Prerequisites for using Deep Learning to solve our problem
  - Large volume of data (hundreds of images for one category)
  - Expertise to design deep architecture and tune hyper-parameters
  - Great computation power to do the training
  - The expertise is usually not revealed in the released paper (kind of secret)
- The theory behind Deep Learning is not well-developed
  - Many theories like dropout lack mathematical proof (trick)
  - Lacks introspection into the features extracted
  - Intuition + Experience + Patience
- Combination of ConvNet + Recurrent Neural Network (LSTM) is powerful for detection problem
  - We have proved that CNN + LSTM would solve the notorious occlusion in detection

Conclusion (cont.)

SIEMENS

- Using trained models is the trend
  - Many trained models are available (Googles' Inception v3)
  - Fine tune the model using our data instead of train from scratch
  - Slightly modify their architecture to meet our need (add or remove layers) or concatenate to other architecture like Recurrent Neural Network

Future Work

Restricted © Siemens AG 2016. All rights reserved

Future Work

SIEMENS

- Do detection by attaching a classification head (FCC + softmax) to the ConvNet
  - Use LSTM for box coordinates regression
  - Use softmax for classification
- Deploy our model using Tensorflow Serving (gRPC + docker)
  - Run our objects detection engine on cloud service
  - Easy to plug the model as component into NX or TeamCenter
- Collect a data set for validation
  - All of our data are NX generated (statistically different from natural images)
  - 3D print the NX mechanic parts
- Investigate into making an similar image search engine
  - Most image search engine are using pixel level comparison or color distribution are obsolete
  - Based on high level features extracted from ConvNet
- Investigate the R-FCN architecture which the newest and claiming to be the best in objects detection [15]

SIEMENS

References

Restricted © Siemens AG 2016. All rights reserved

References

SIEMENS

- [1] Stanford CS231n: Convolutional Neural Networks for Visual Recognition
- [2] Stanford CS224d: Deep Learning for Natural Language Processing
- [3] Jiuxiang Gu et al. Recent Advances in Convolutional Neural Networks
- [4] Alex Krizhevsky et al. ImageNet Classification with deep convolutional neural networks (NIPS 2012)
- [5] K. Simonyan & A. Zisseraman. Very deep convolutional networks for large-scale image recognition (ICLR 2015)
- [6] Christian Szegedy et al. Rethinking the Inception architecture for computer vision
- [7] Yoshua Bengio. Practical Recommendation for Gradient-Based Training of Deep Architectures (2012)
- [8] C. Smith, C. Nguyen et al. Distributed TensorFlow: Scaling Google's Deep Learning Library on Spark (Spark Summit 2016)
- [9] P. Moritz et al. SparkNet: Training Deep Neural Network in Spark
- [10] R. Girshick et al. Rich feature hierarchies for accurate object detection and semantic segmentation
- [11] Ross Girshick. Fast R-CNN. Microsoft Research
- [12] S. Ren et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
- [13] Pierre Sermanet et al. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks
- [14] R. Stewart et al. End-to-end people detection in crowded scenes
- [15] Jifeng Dai et al. R-FCN: Object Detection via Region-based Fully Convolutional Networks
- [16] M. Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems

References (cont.)

SIEMENS

- [17] http://neuralnetworksanddeeplearning.com/chap4.html
- [18] http://neuralnetworksanddeeplearning.com/chap5.html
- [19] https://github.com/tensorflow/models/tree/master/inception
- [20] colah.github.io
- [21] http://www.amax.com/blog/?p=804
- [22] https://github.com/ethereon/caffe-tensorflow

SIEMENS

Q & A

Restricted © Siemens AG 2016. All rights reserved

Contacts

Zhenyu Wu: wu.2880@osu.edu

Neeraj Panhalkar: neeraj.panhalkar@siemens.com

Restricted © Siemens AG 2016. All rights reserved