Electrical Engineering & Computer Science

OSU Oregon State University

COLLEGE OF ENGINEERING

Some Thoughts and New Designs of Convolutional and Recurrent Architectures

Fuxin Li

JUNE 7TH 2019

The Deep Learning Era

ImageNet Classification

Go-Playing

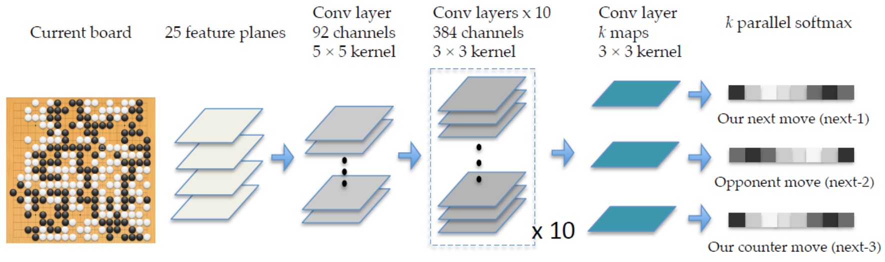
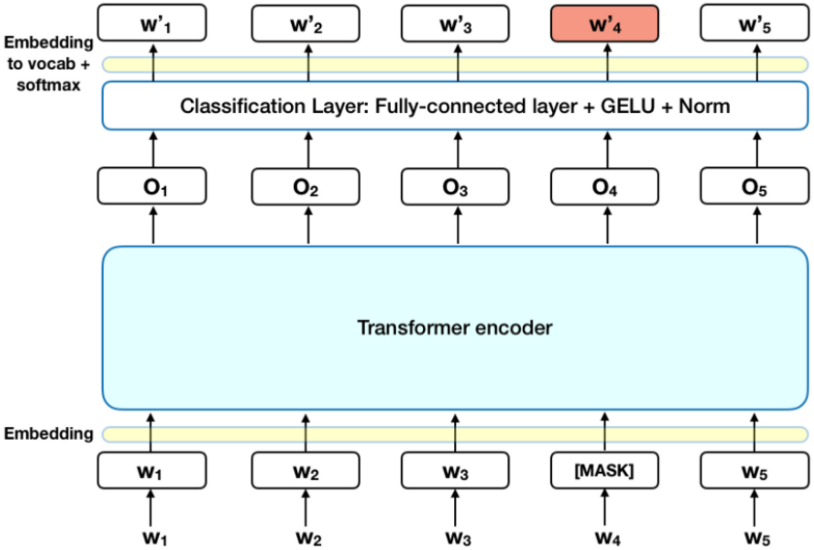
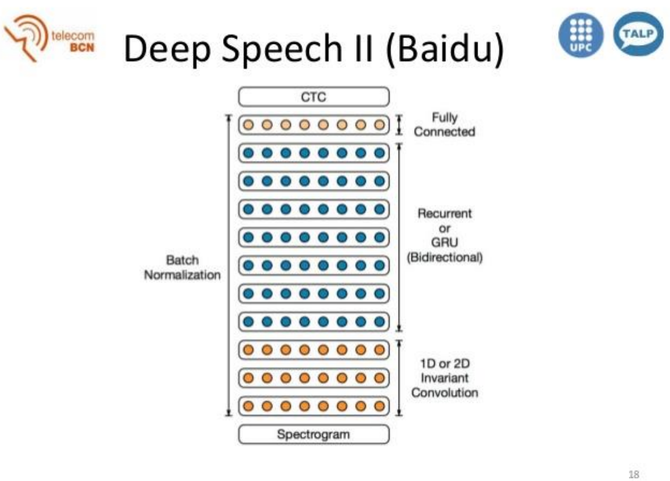


Figure 3: Our network structure ($d = 12$, $w = 384$). The input is the current board situation (with history information), the output is to predict next k moves.

Natural Language Processing



Speech Recognition



Academics in the Deep Learning Era

- Challenging time for academics…
  - Deep learning depends heavily on GPUs and data
  - Universities don't possess a lot of GPUs and data

Analysis

AlphaGo Zero learns to play Go by simulating matches against itself in a procedure referred to as **self-play**. The paper reports the following numbers:

At the quoted rate of **$6.50/TPU/hr** (as of March 2018), the whole venture would cost **$2,986,822** in TPUs alone to replicate. And that's just the smaller of the two experiments they report:

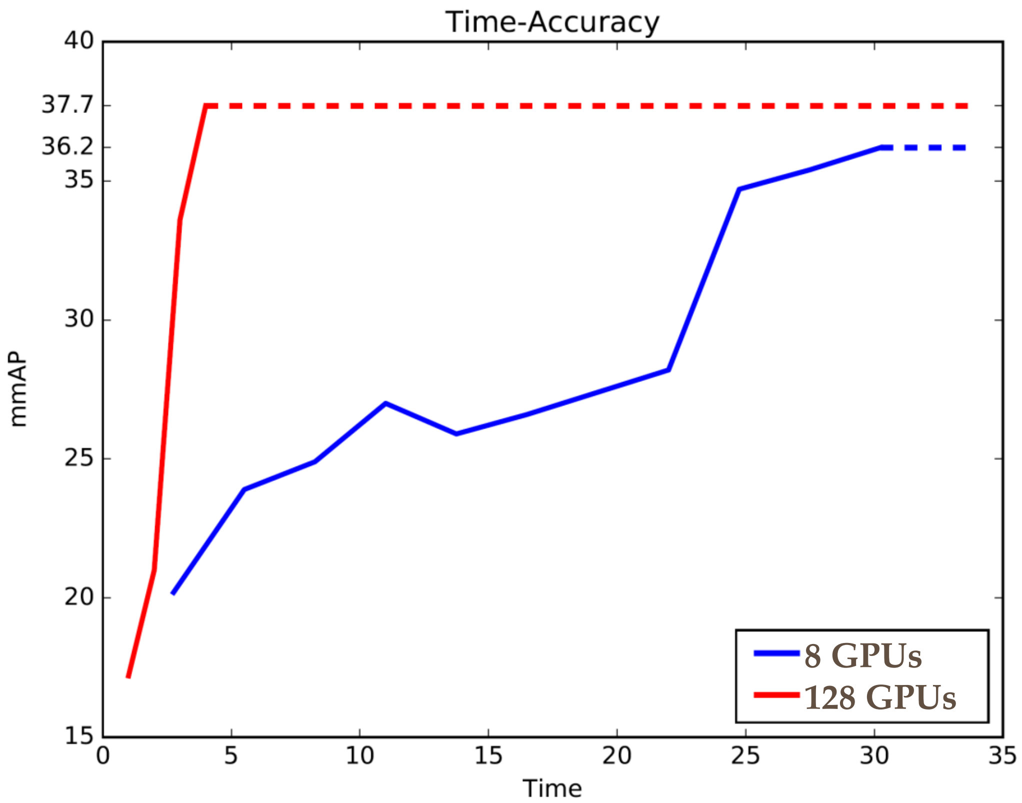
NASNet: Neural Architecture Search with Reinforcement Learning

Training details: The controller RNN is a two-layer LSTM with 35 hidden units on each layer. It is trained with the ADAM optimizer (Kingma & Ba, 2015) with a learning rate of 0.0006. The weights of the controller are initialized uniformly between -0.08 and 0.08. For the distributed training, we set the number of parameter server shards S to 20, the number of controller replicas K to 100 and the number of child replicas m to 8, which means there are 800 networks being trained on 800 GPUs concurrently at any time.

Zopf & Le. ICLR 2017

Academics in the Deep Learning Era

- Challenging time for academics…
  - Deep learning depends heavily on GPUs and data
  - Universities don't possess a lot of GPUs and data



cf. Peng et al. MegDet: A Large Mini-Batch Object Detector. CVPR 2018

Deep Networks: Understandings and New Paradigms

- As academics, we choose a different route

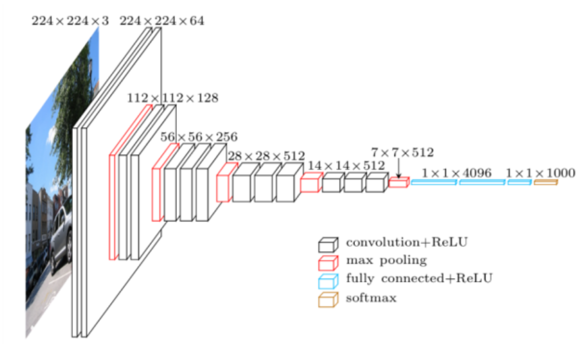
Paradigm/Understanding > Performance

We want to study the deep networks themselves

Extend our understanding and propose new paradigms

Roadmap for Today

CNN

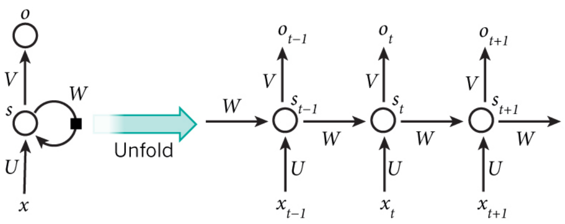


Explaining Predictions

Quantifying Uncertainty

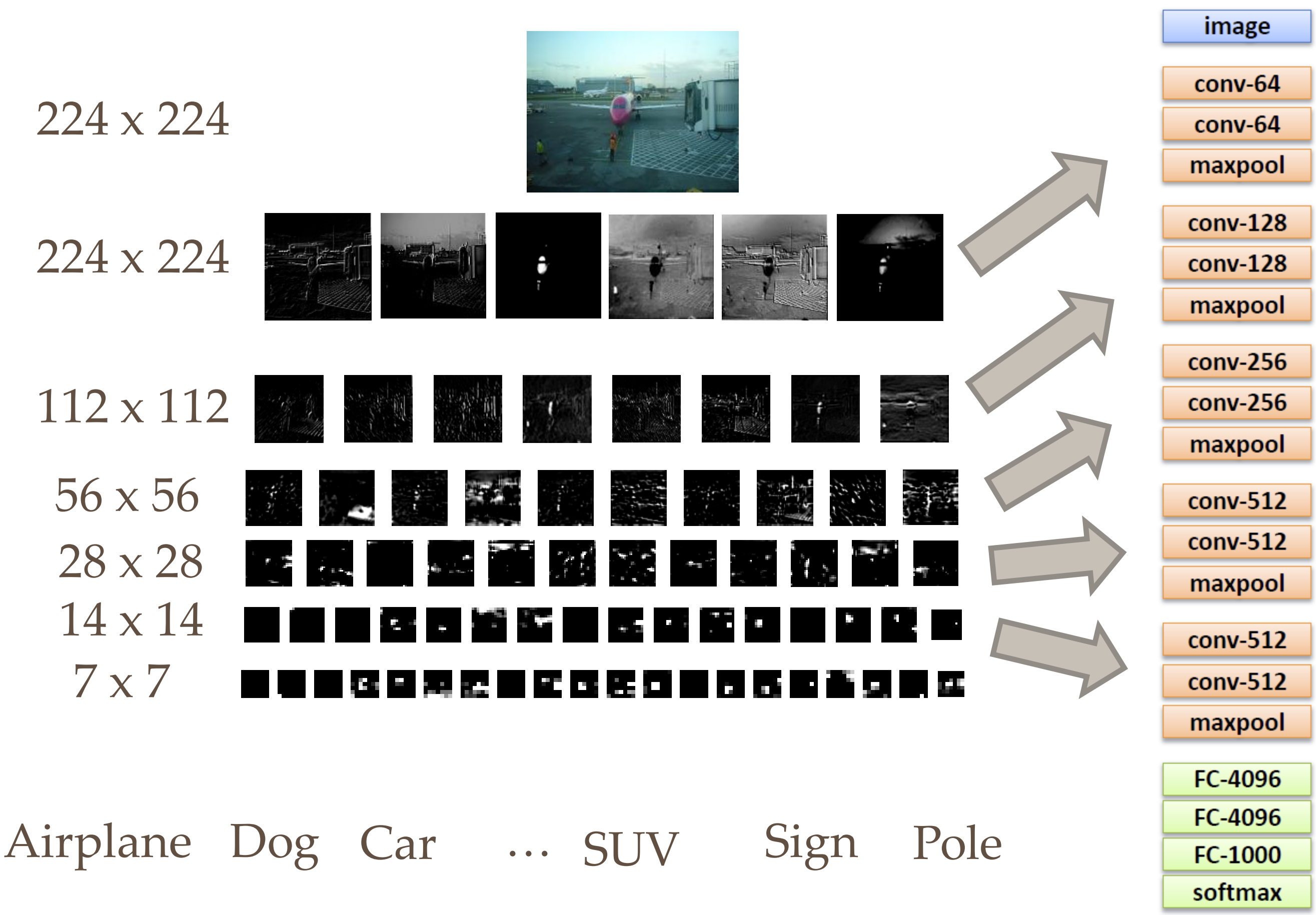
CNN on Unordered Point Clouds

RNN



Memory Design for Multi-Target Tracking

Understanding Convolutional Networks (CNN)



(Simonyan and Zisserman 2014)

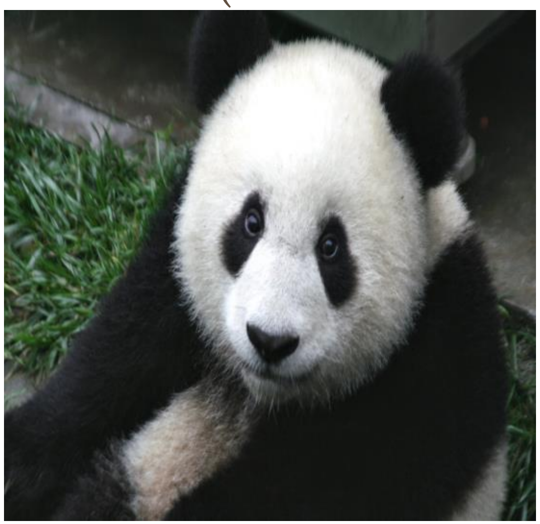
Fooling a deep network(Szegedy et al. 2013)

Optimizing a delta from the image to maximize a class prediction $f_c(x)$

$$\max_{\Delta I} f_c(I + \Delta I) - \lambda||\Delta I||^2$$

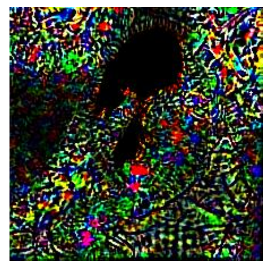
Giant Panda (99.32% confidence)

I



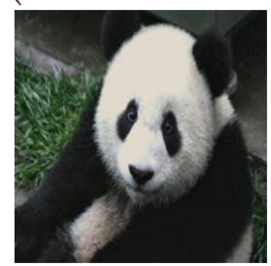
$+0.03$

ΔI



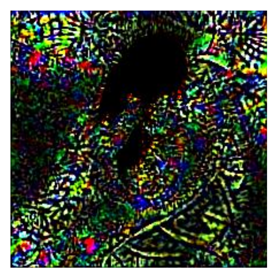
$=$

Shark (93.89% confidence)



$+0.03$

ΔI



$=$

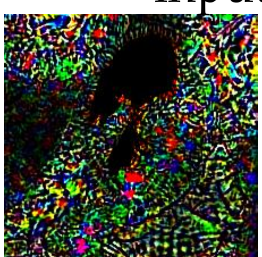
Goldfish (95.15% confidence)

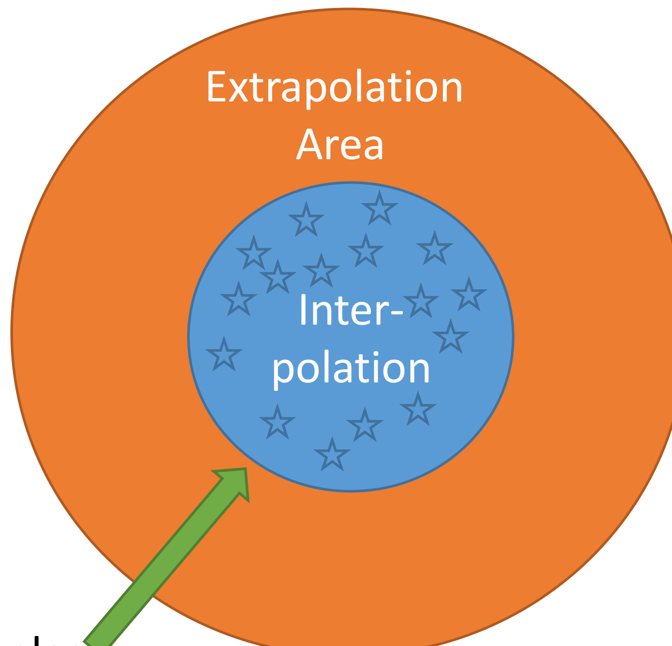
(Szegedy et al. 2013, Goodfellow et al. 2014, Nguyen et al. 2015)

Fundamental Aspect of ML

- Machine learning works only on i.i.d. settings
  - Testing data should be similar to training data
  - No good result expected on adversarial images since never trained on it
  - CNN tends to give random outputs with high confidence

These are no longer within the input distribution!

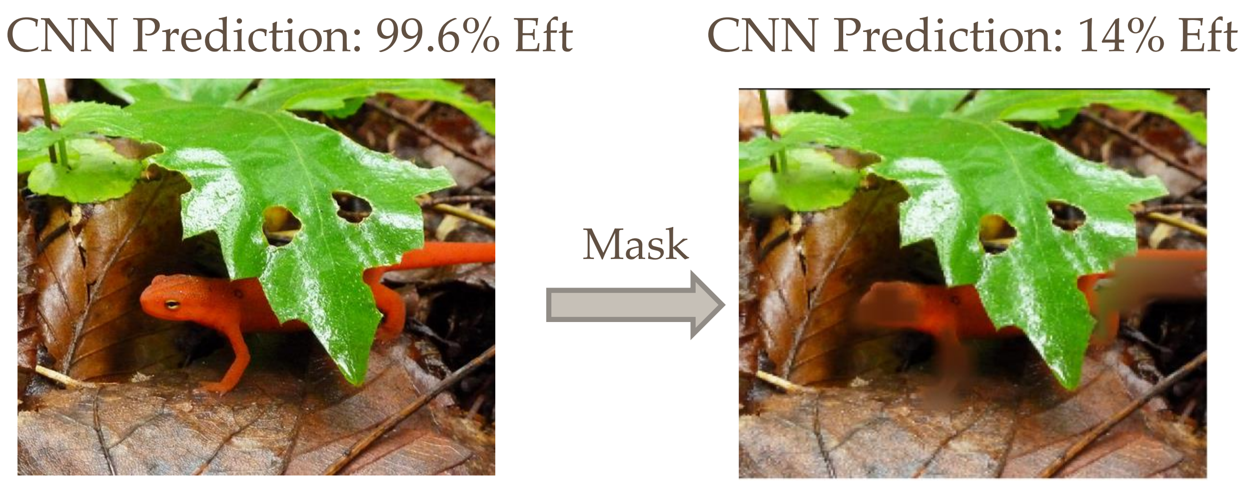




I-GOS: Integrated-Gradient Optimized Saliency

Optimize for a mask that will mask an image to its highly blurred version

Locate the smallest **non-adversarial** mask to reduce prediction score



Masking Optimization

I_0 is the image
M is the mask
$f_c(x)$ is the deep network

$$\underset{M}{\operatorname{argmin}} \; F_c(I_0, M) = f_c\big(\Phi(I_0, M)\big) + g(M),$$

$$\text{where} \;\; g(M) = \lambda_1 ||\mathbf{1} - M||_1 + \lambda_2 \mathrm{TV}(M),$$

$$\Phi(I_0, M) = I_0 \odot M + \tilde{I}_0 \odot (\mathbf{1} - M),$$

$$\mathbf{0} \leq M \leq \mathbf{1},$$

Avoiding Local Optima

Just using gradient ->
Easily falling to local optima

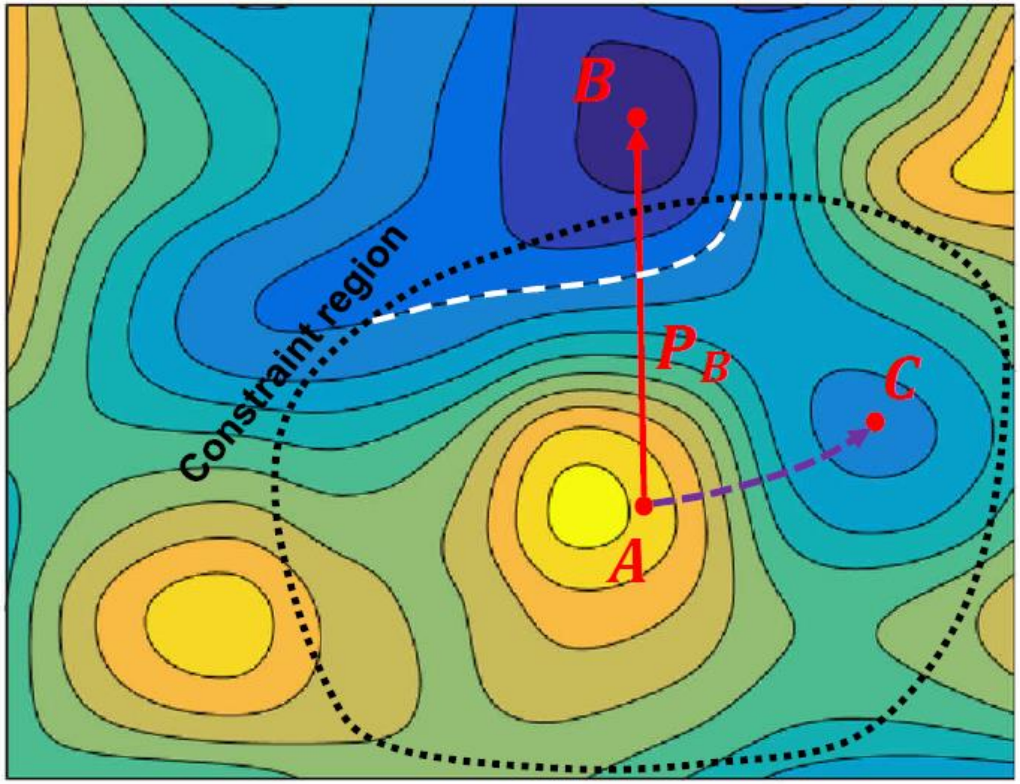
Try to get to global optima

Highly blurred image
= unconstrained
global optima (outputting
0 confidence)

We have constraints!

Masks need to be small

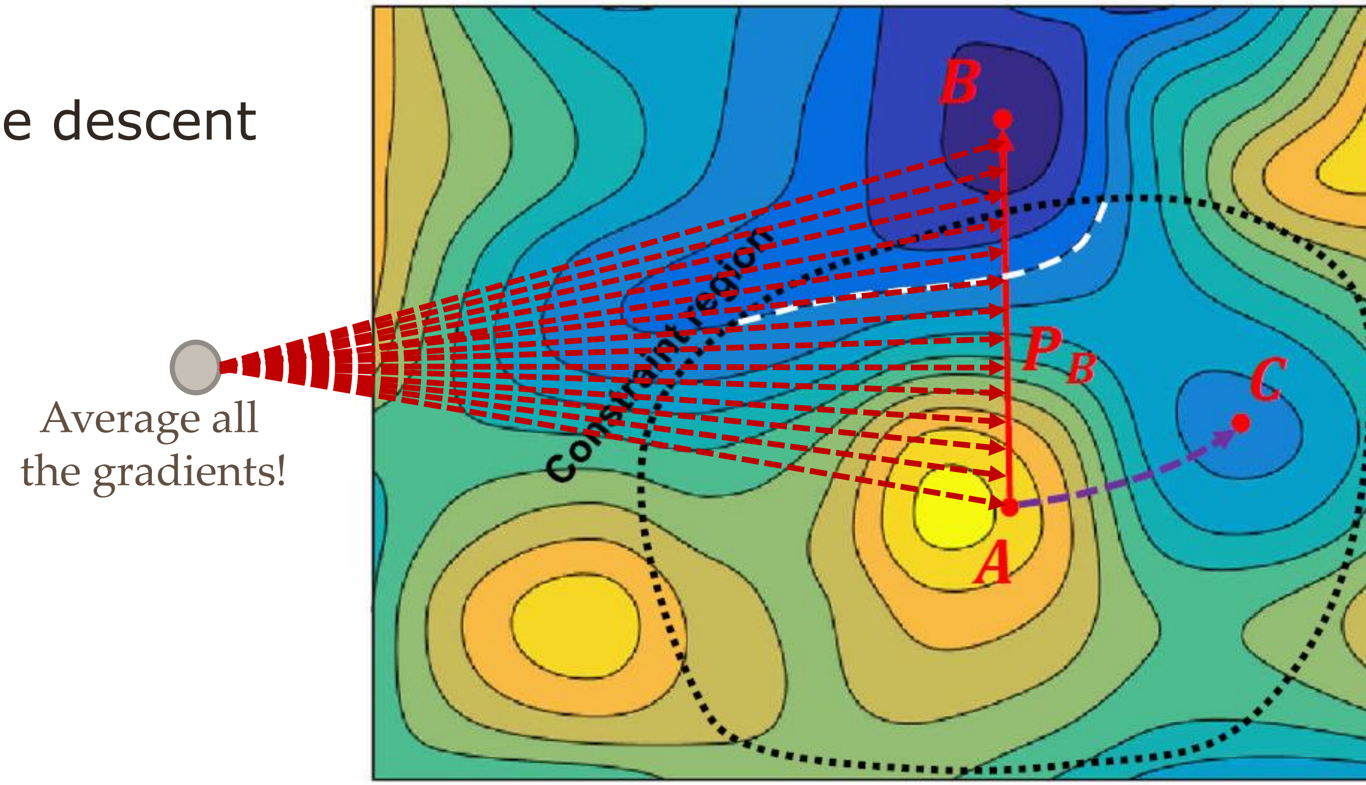
Low total variation on mask



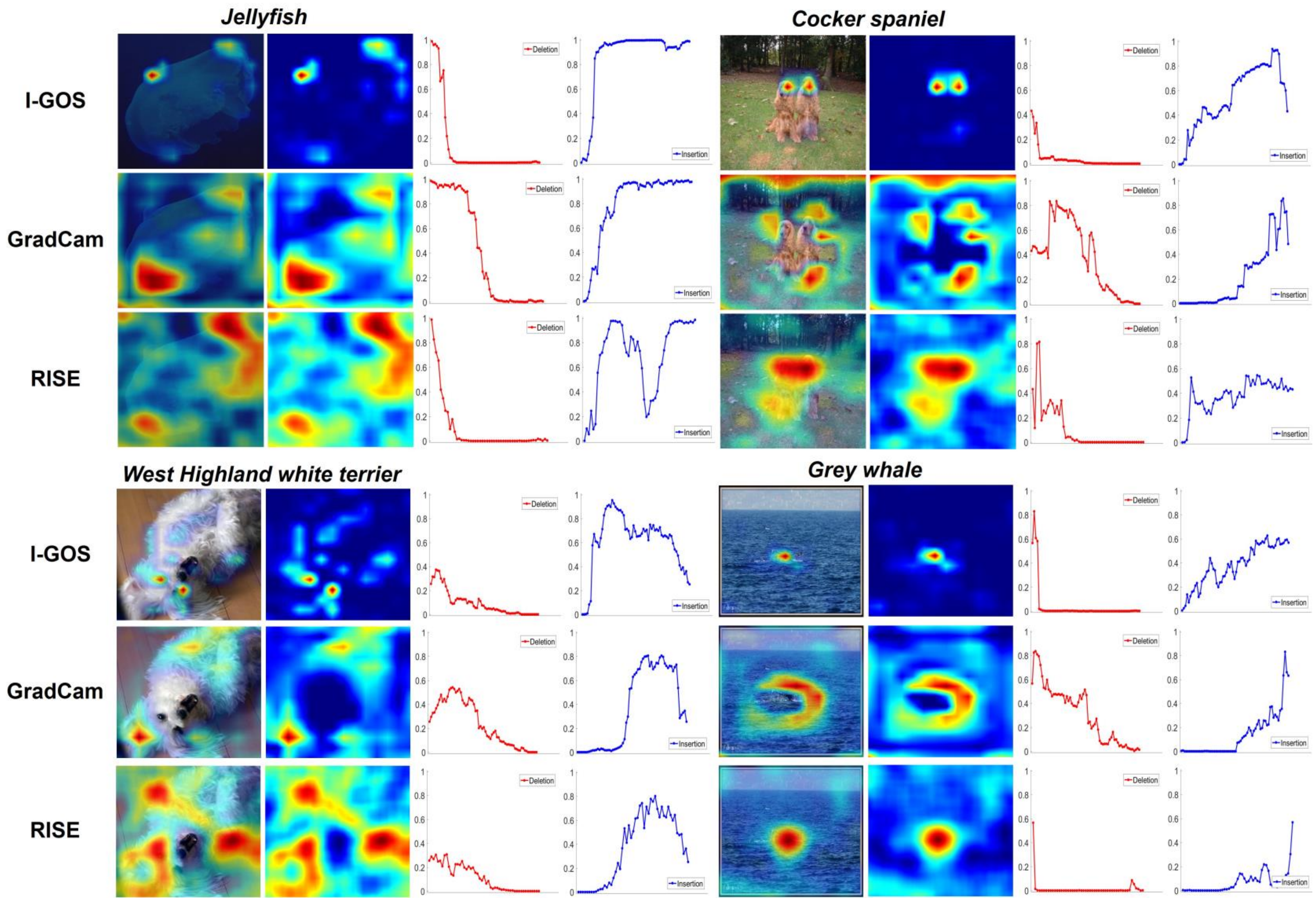
Integrated Gradient

Take gradients at many locations on the line A->B

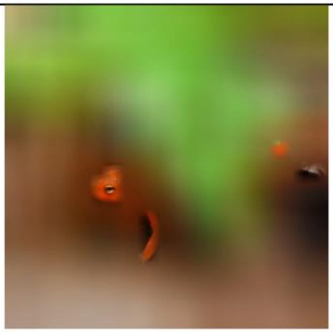
Use that as the descent step



Comparing I-GOS with Other Visualizations

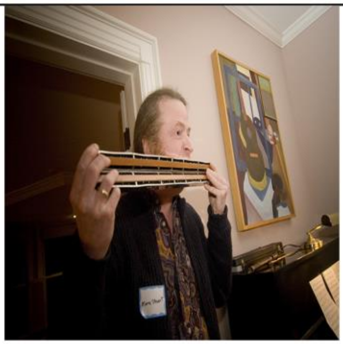
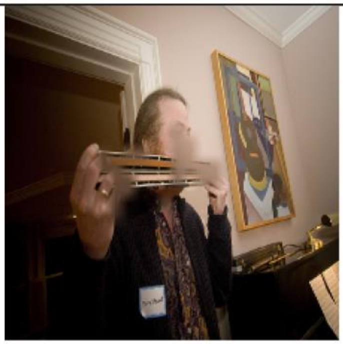
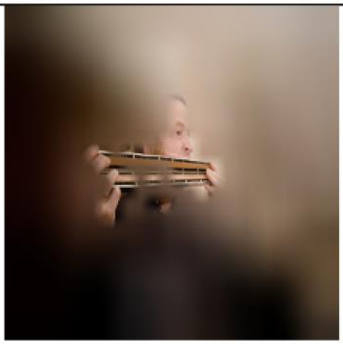


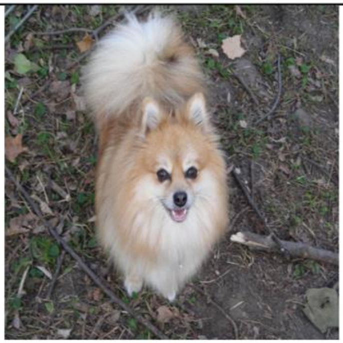
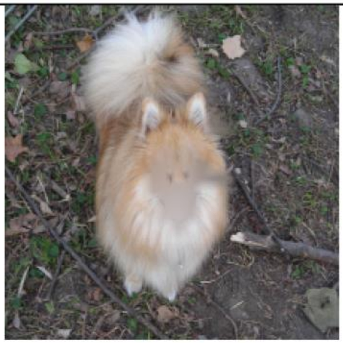
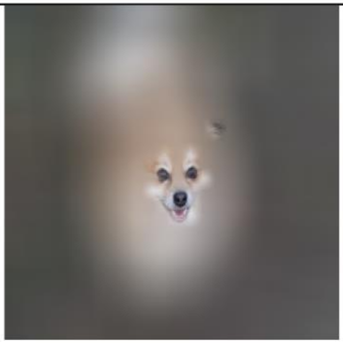
CNN diagnosis: Generating Images from the mask

Label	Original Image	Deletion	Insertion
27: 'Eft'			
Predicted Class Probability	99.6%	14%	97%
Deletion or Insertion ratio	– –	6.1%	1.5%

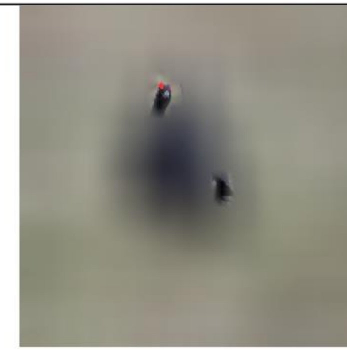
Label	Original Image	Deletion	Insertion
409: 'Analog clock'			
Predicted Class Probability	34.1%	4.3%	35.5%
Deletion or Insertion ratio	– –	1.9%	0.8%

CNN diagnosis: Generating Images from the mask

Label	Original Image	Deletion	Insertion
593: 'Harmonica, mouth organ, harp, mouth harp'			
Predicted Class Probability	99.9%	11.9%	81.8%
Deletion or Insertion ratio	– –	3.1%	4.6%

Label	Original Image	Deletion	Insertion
259: 'Pomeranian'			
Predicted Class Probability	100%	4.8%	82.9%
Deletion or Insertion ratio	– –	3.4%	2.3%

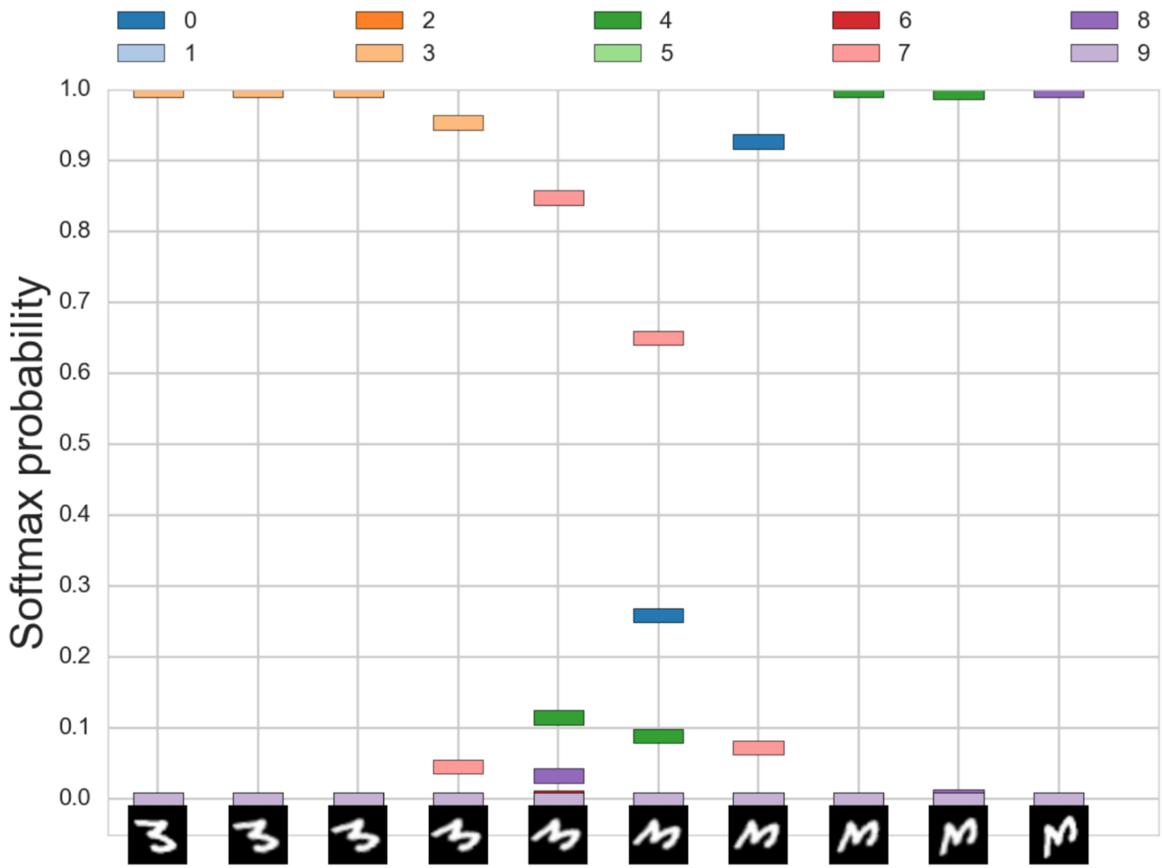
CNN diagnosis: Generating Images from the mask

Label	Original Image	Deletion	Insertion
80: 'Black grouse'			
Predicted Class Probability	99.5%	0.9%	99.7%
Deletion or Insertion ratio	– –	2.7%	0.8%

Label	Original Image	Deletion	Insertion
437: 'Beacon, lighthouse, beacon light, pharos'			
Predicted Class Probability	95.7%	12.3%	78.1%
Deletion or Insertion ratio	– –	0.8%	1.5%

Uncertainty in Deep Learning

A deep network is almost always overconfident in its prediction



Images cf. Louizos & Welling 2017

Uncertainty in Deep Learning

- How to correct this overconfidence on outliers?
  - A Bayesian Idea:

$$\boldsymbol{p}(\boldsymbol{y}|\boldsymbol{X}) = \int \boldsymbol{p}(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{W})\boldsymbol{q}(\boldsymbol{W})\boldsymbol{d}\boldsymbol{W}$$

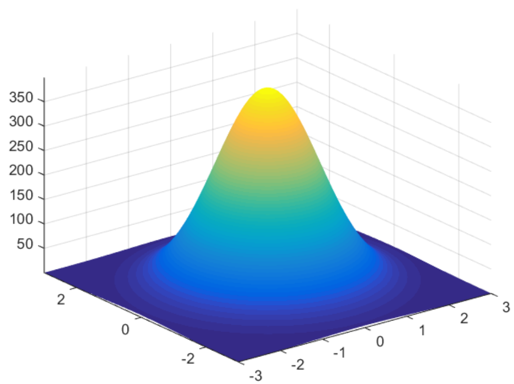
  - Given a prior distribution $p(W)$, learn the posterior $q(W)$ so that many models can be sampled
  - Presumably, different models predict ***similarly*** on inliers but ***differently*** on outliers
    - Outliers would have higher predictive entropy

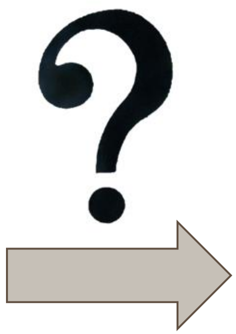
How to Get Multiple Models

- Train an Ensemble (Zeiler & Fergus 2012, Lakshminarayanan et al. 2016)
  - Too slow
- MC-Dropout (Gal & Ghahramani 2016)
  - Uses dropout to simulate an ensemble
- Multiplicative Normalizing Flow (MNF) (Louizos & Welling, 2017)
  - Use a normalizing flow to compute posterior
  - Hard to scale
  - Assume multiplicative Gaussian noise on weights

Problem: Distribution Assumptions Too Restrictive

- We know too few distributions to tractably compute priors/posteriors
  - Only normal, exponential, etc.





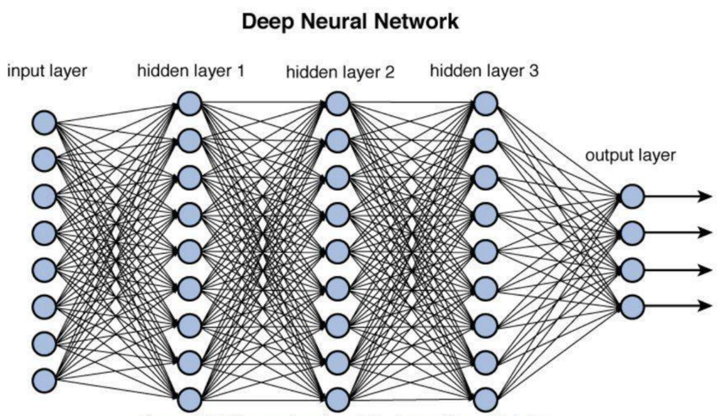


Figure 12.2 Deep network architecture with multiple layers.

- e enoug
  of deep network parameters!

Directly Generate a Neural Network?

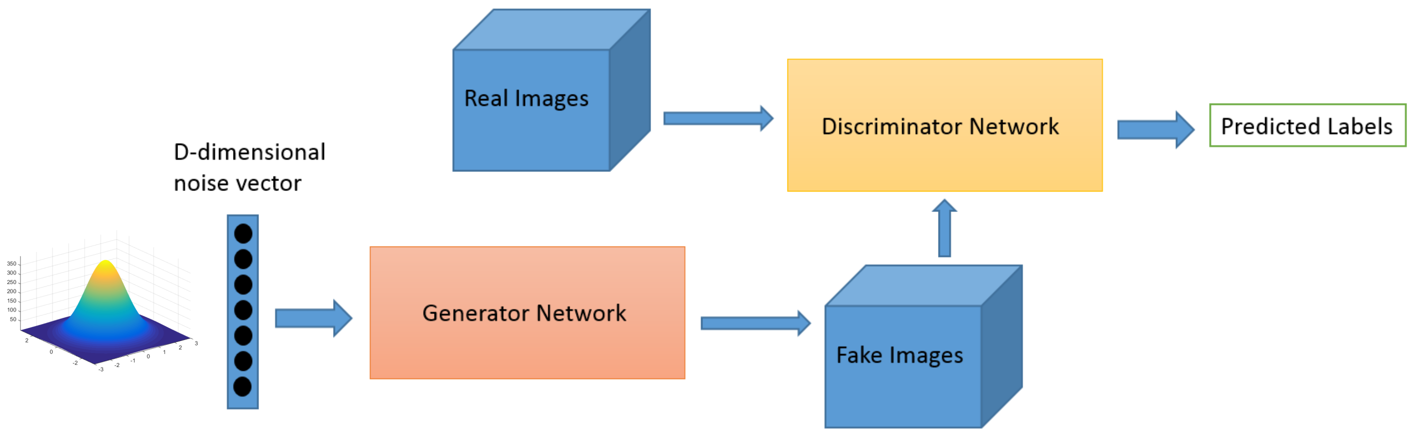
- GANs are known to be good generators on arbitrary distributions
  - E.g. Images

- Can we use a GAN to generate a neural network?
  - Given structure, generate all the weights

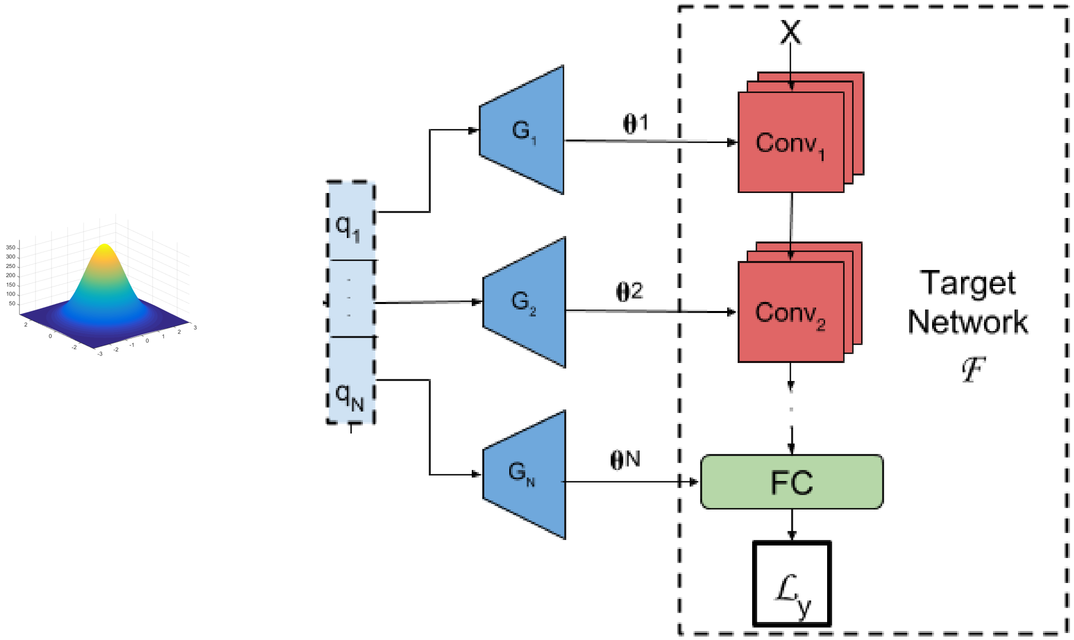
Images cf. Karras et al. 2018, Brock et al. 2019

A GAN Generated Neural Network?

GAN:

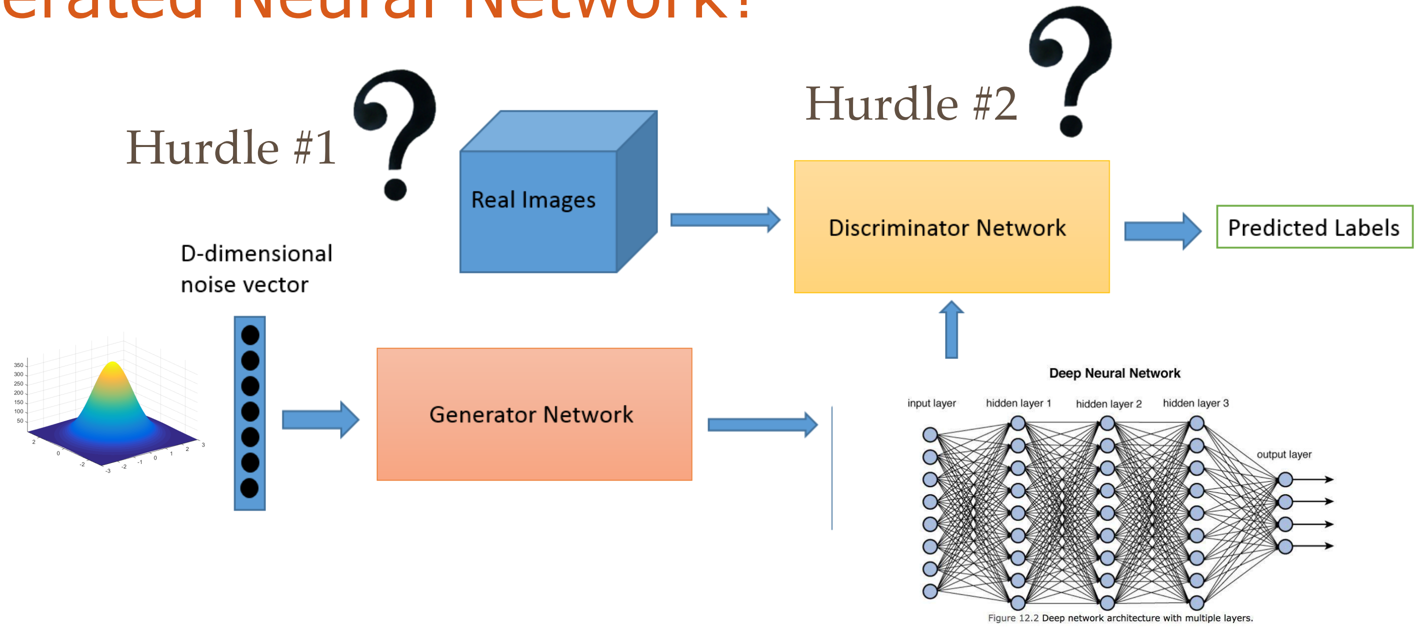


Generating a Neural Network:



A GAN Generated Neural Network?

GAN:



- Two big hurdles:
  - Real data – Train 1,000 networks first? (APD, Wang et al. 2018)
  - Adversarial loss – Discriminator on network weights?
    - No structure to utilize as in images

Image cf. O'Reilly

Solution #1: Maximize Likelihood

- Assume unknown true parameter distribution θ^* in a parametric model, the KL-MLE equivalence is well known:

$$\min_{\theta} D_{KL}[p(x,y|\theta^*)||p(x,y|\theta)]$$

$$\Leftrightarrow \min_{\theta} \left[\mathbb{E}_{x,y}[\log p(x,y|\theta^*)] - \mathbb{E}_{x,y}[\log p(x,y|\theta)]\right]$$

$$\Leftrightarrow \max_{\theta} \mathbb{E}_{x,y}[\log p(y|x,\theta)]$$

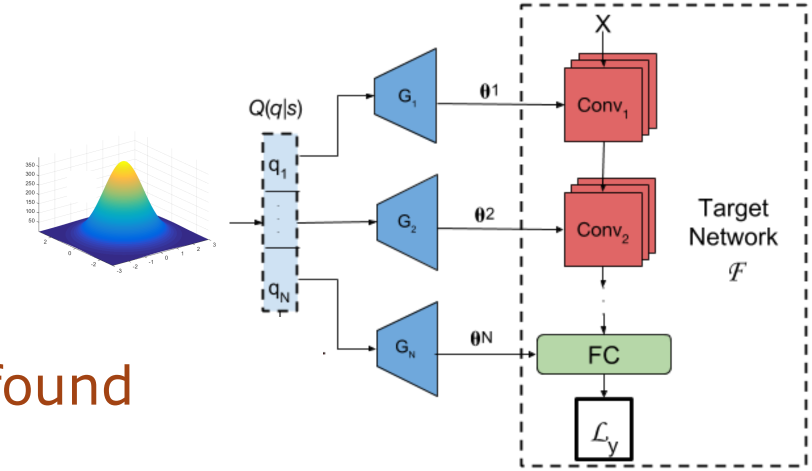
Using MLE is equivalent as minimizing a "reconstruction error" for generating $\boldsymbol{\theta}$

Solution #2: Latent Space Discriminator

- Wasserstein autoencoder (Tolstikhin et al. 2018) justified the validity of latent space discriminator
  - Instead of a discriminator from data (images), train a discriminator from the latent codes
- Adopting this solved the discriminator issue
  - Latent space is much lower dimensional than network parameters

Training Issue:

- We found the initial architecture hard to train
  - Generator has to figure out:
  - Next layer's input is prev layer output

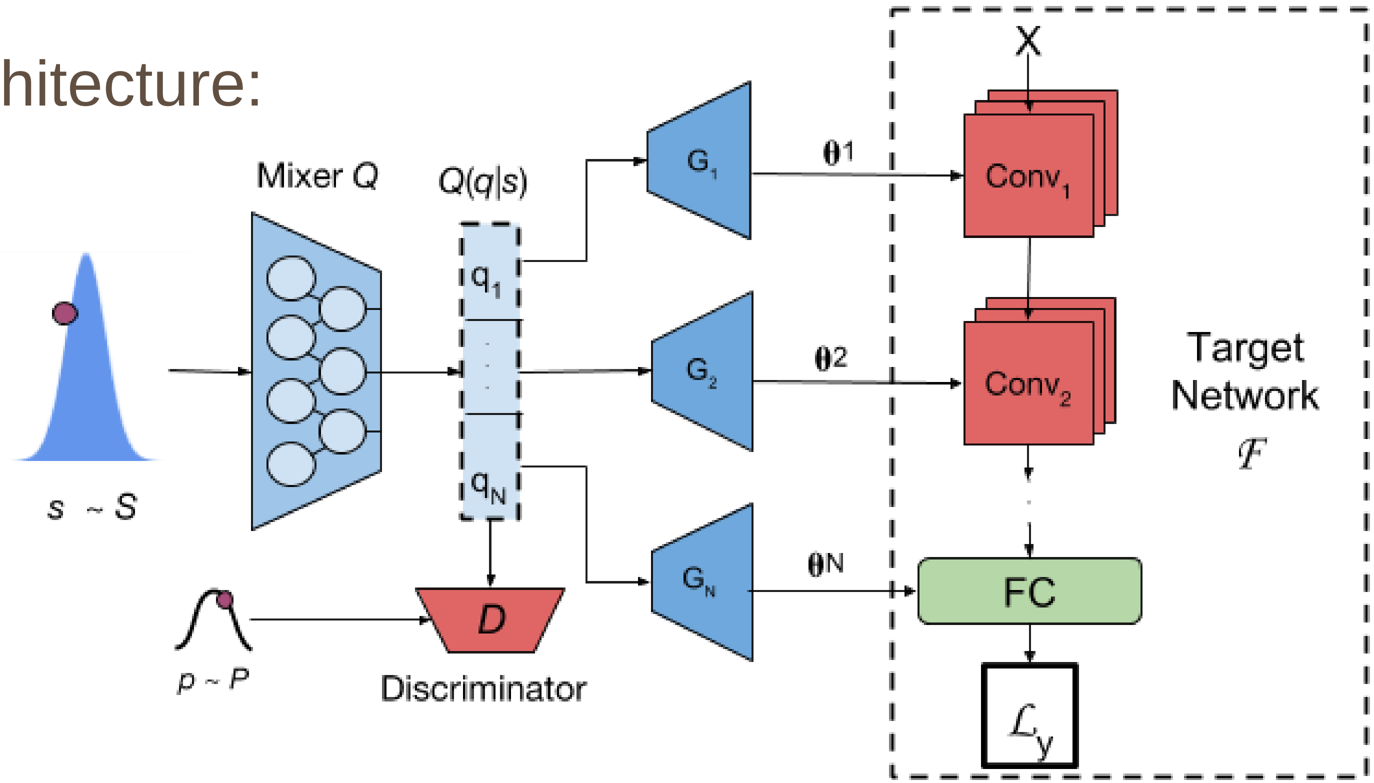


- This is not easy
  - It leads to mode collapse
  - Only 1 good network can be found

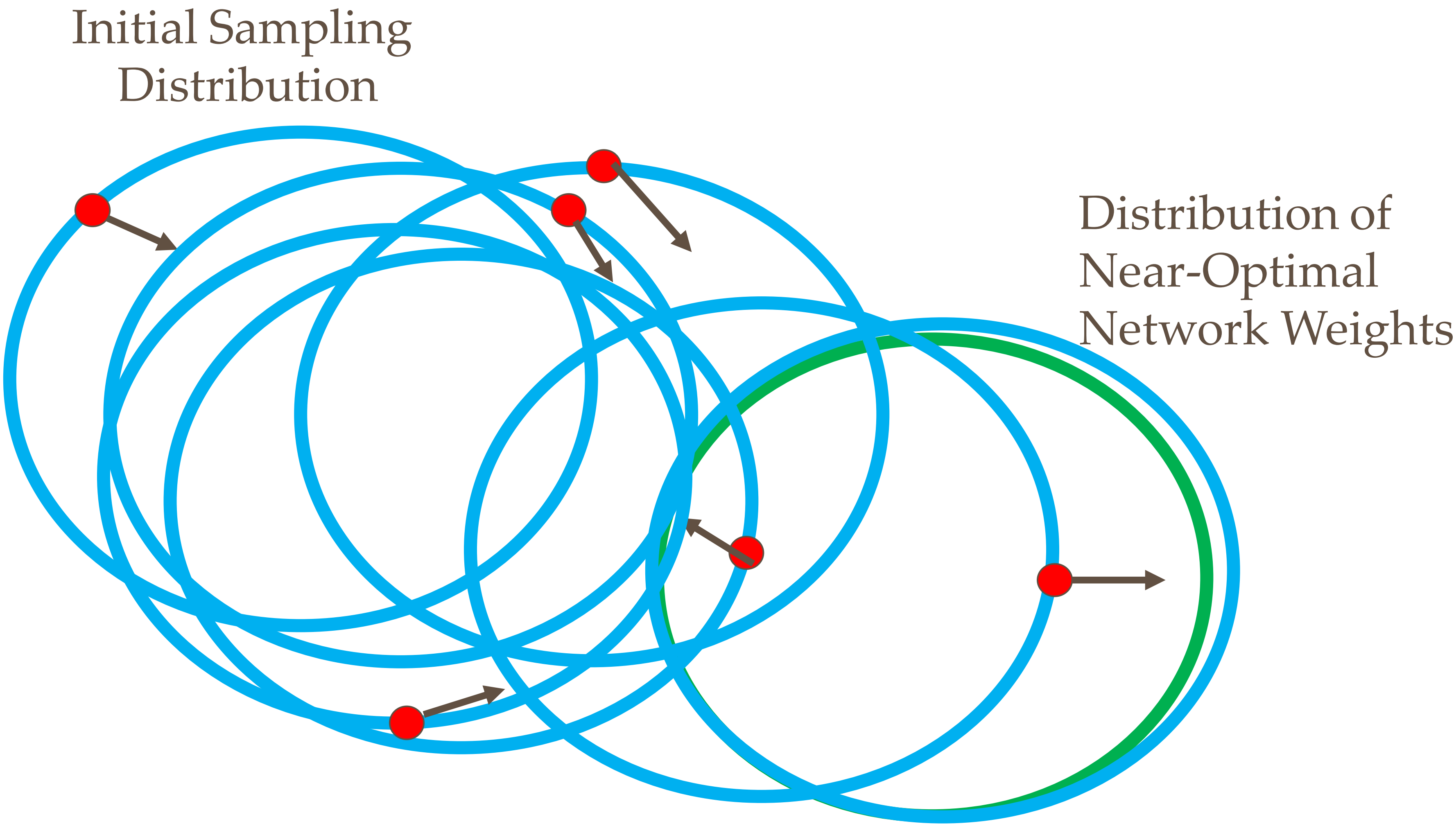
Novel Mixer

- We add a mixer to “mix” the independent Gaussian noise
  - Different generators have dependent info to work on

Final Architecture:

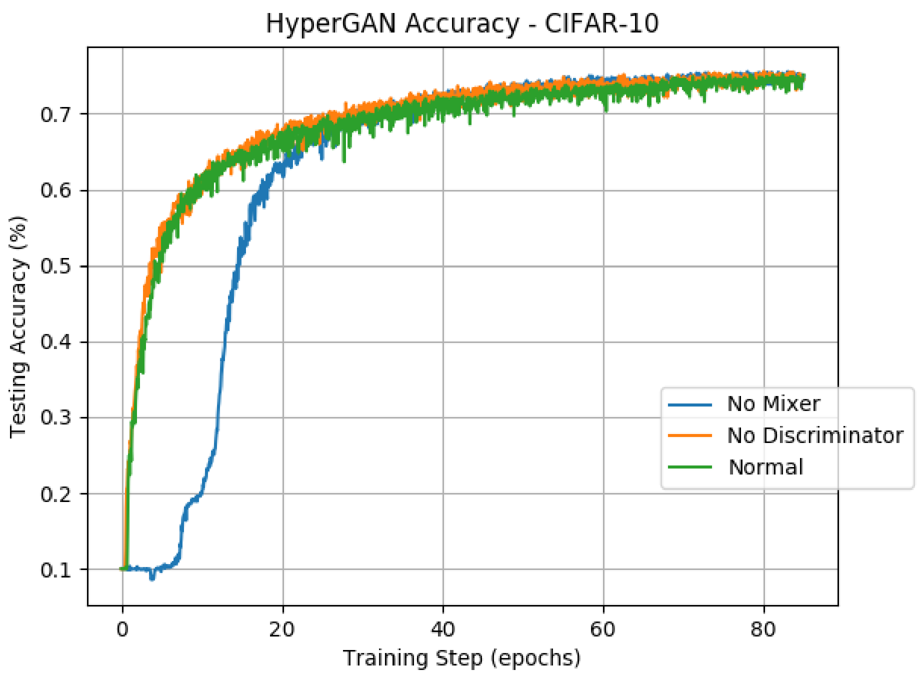


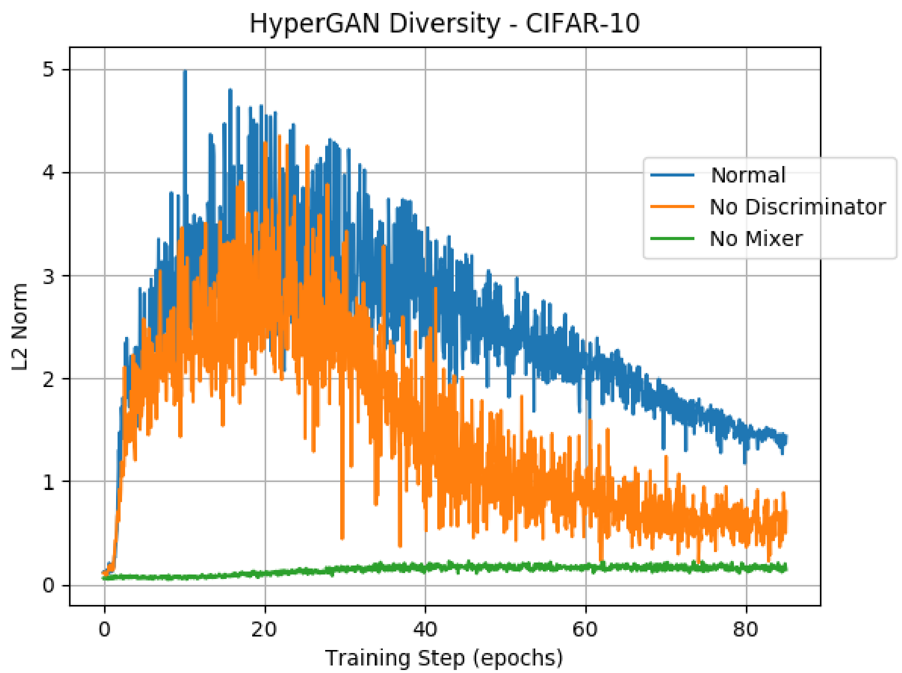
Training Illustration



Mixer and Discriminator Encourage Diversity

- Accuracy is about the same in any case
- No mixer = almost no diversity
- No discriminator = less diversity





Experiments: Classification Accuracy

- Baselines: 100 network ensembles from APD, MNF (note: small network since MNF doesn't scale)

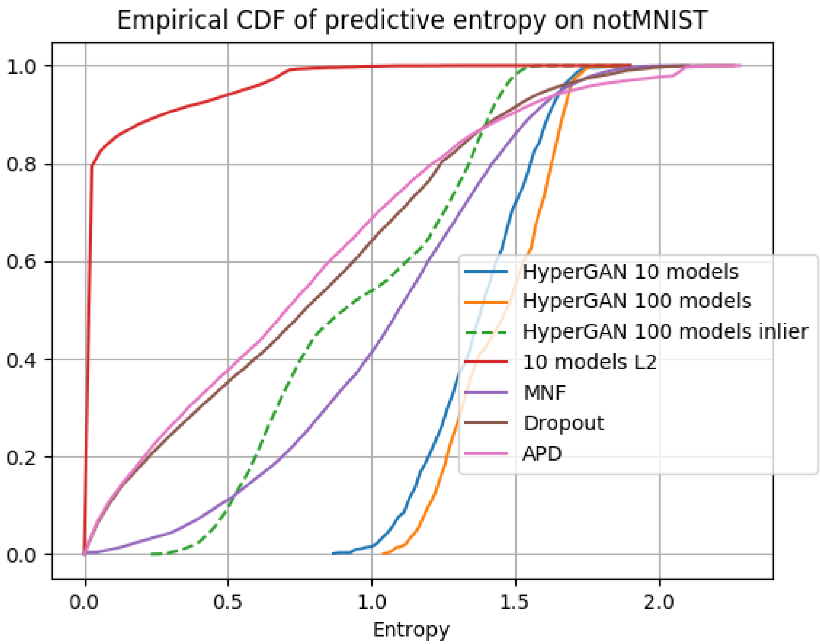
Method	MNIST	MNIST 5000	CIFAR-5	CIFAR-10	CIFAR-10 5000
1 network	98.64	96.69	84.50	76.32	76.31
5 networks	98.75	97.24	85.51	76.84	76.41
10 networks	99.22	97.33	85.54	77.52	77.12
100 networks	**99.31**	**97.71**	**85.81**	**77.71**	**77.38**
APD	98.61	96.35	83.21	75.62	75.13
MNF	99.30	97.52	84.00	76.71	76.88
MC Dropout	98.73	95.58	84.00	72.75	70.10

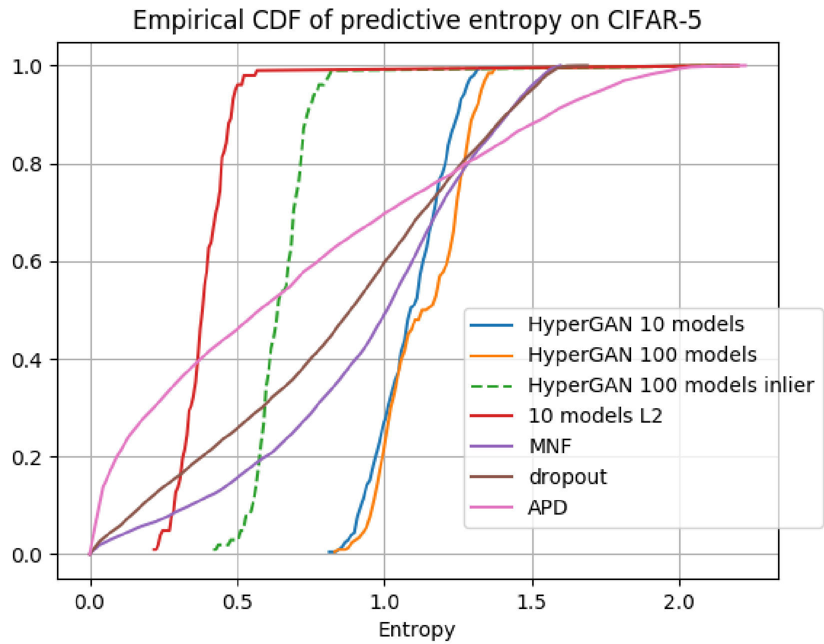
Diversity helps!

MNIST/CIFAR 5000: train on 5k example subset
CIFAR-5: 5 classes out of CIFAR-10

Experiments: Outlier Detection

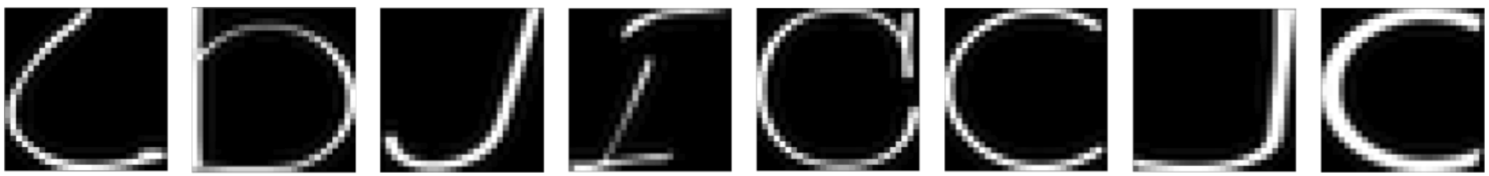
- Train on MNIST -> Predict on notMNIST (characters)
- Train on CIFAR 5 -> Predict on other 5 classes





Outlier Examples

notMNIST low entropy examples:

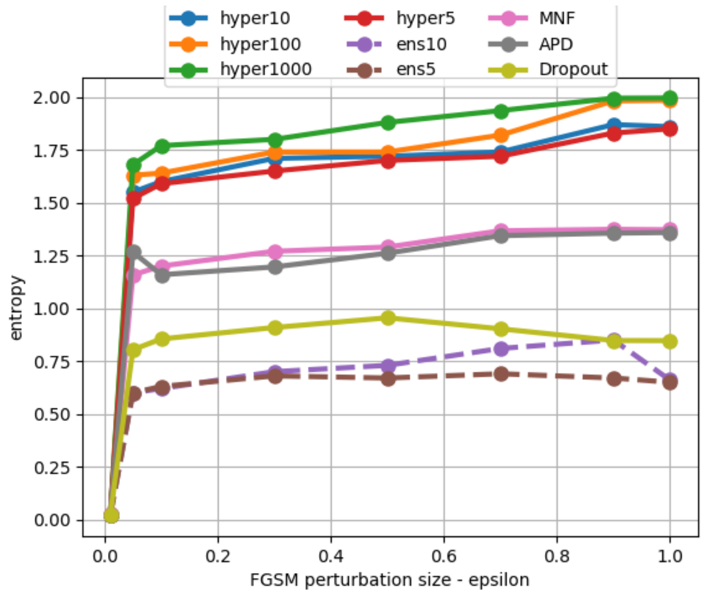


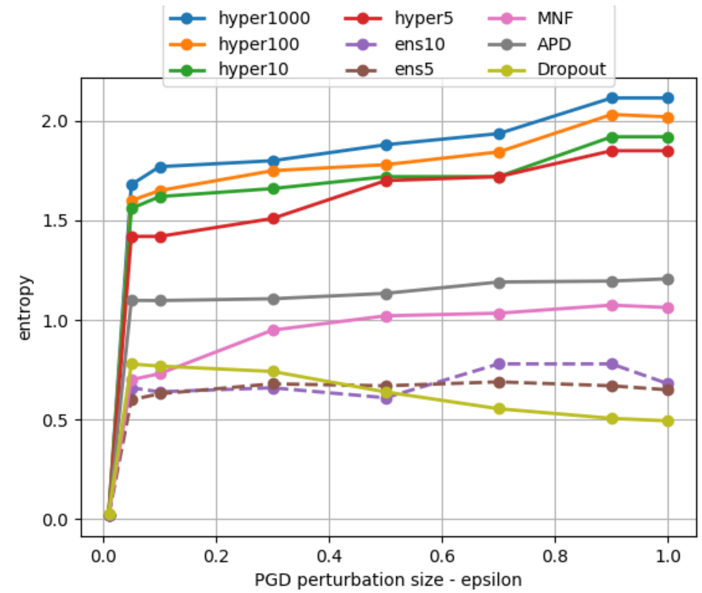
MNIST high entropy examples:

Adversarial Examples

- Problem Setup: Shifting Classifier:
  - Adversaries fool one ensemble
  - HyperGAN generates a new ensemble of same size

Evaluate predictive entropy: higher means less fooled



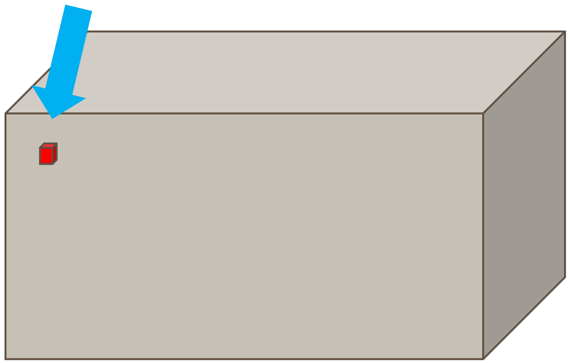


Deep Learning in 3D Vision

- Processing 3D (laser scan, RGB-D) data is important
- Many robotic applications have sensors to directly collect 3D data
- Radar, sonar applications have direct 3D data

- But Deep CNN is prohibitively expensive in 3D
- Some work utilizes efficient data structures (e.g. octrees) to speed up
- But still not enough

Rasterized Deep CNN:
Every voxel needs to be processed



W. Wu, Z. Qi, **FL**, arXiv:1811.07246

Point Cloud Data

- Directly representing each point on a surface
- Avoid representing unoccupied points
- Can be economical and extremely accurate
- Difficulty: Irregular data is hard to process!
- Goal: Build CNN-like networks on point clouds directly

W. Wu, Z. Qi, **FL**, arXiv:1811.07246

Prior Work on Point Cloud Networks

- PointNet (Qi et al. 2017)
- Using max-pooling on points to build a network on point cloud
- PointNet++ (Qi et al. 2017)
- PointNet with a hierarchical structure and local neighborhoods
- Graph Convolution (Simonovsky et al. 2017)
- Treat the point cloud as a graph and perform graph convolution
- SPLATNet (Su et al. 2018), SpiderCNN (Xu et al. 2018), PointCNN (Li et al. 2018)
- Approaches to approximate CNN on point cloud
- None of them are real convolutions!

Convolutions on non-regular neighborhoods

- Directly running CNN on point cloud data
- There is no fixed grid, hence the normal CNN formula does not work

Conventional (Rasterized) CNN:

$$Conv(W,X)_{ijk} = \sum_{(\Delta i,\Delta j,\Delta k)\in G} W_{\Delta i \Delta j \Delta k} X(i+\Delta i, j+\Delta j, k+\Delta k)$$

Approximates the Continuous Domain CNN:

$$Conv(W,X)_{ijk} = \int_{\Delta i,\Delta j,\Delta k} W(\Delta i,\Delta j,\Delta k)\, X(i+\Delta i, j+\Delta j, k+\Delta k)\, d\Delta i\, d\Delta j\, d\Delta k$$

Convolution Function

The Idea: Approximate the convolution function with a neural network!

W. Wu, Z. Qi, **FL**, arXiv:1811.07246

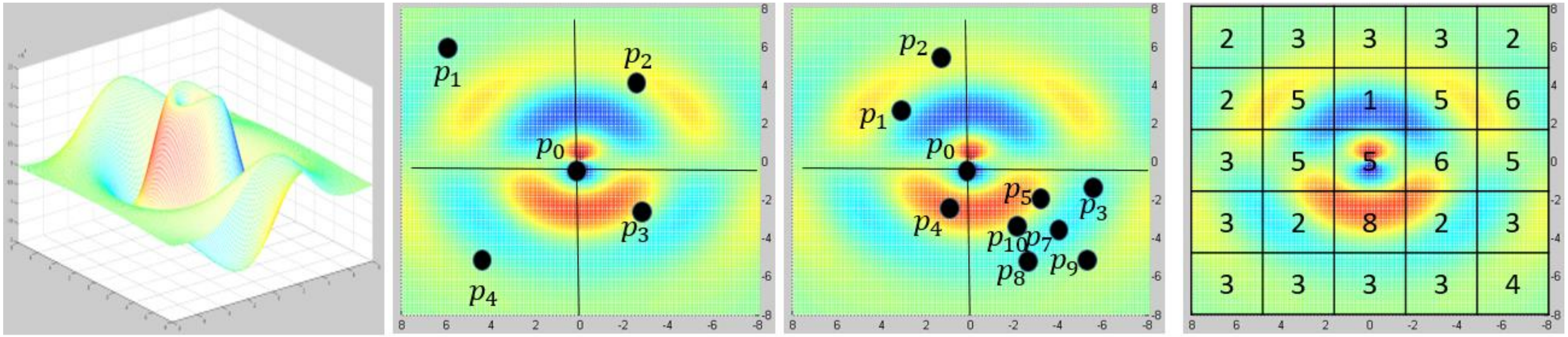
PointConv

- PointConv approximates the continuous convolution operator by:

1-hidden layer neural net

$$PointConv(W,X)_{ijk} = \sum_{(\Delta i,\Delta j,\Delta k)\in G} \frac{1}{d(\Delta i,\Delta j,\Delta k)} W(\Delta i,\Delta j,\Delta k) X(i+\Delta i, j+\Delta j, k+\Delta k)$$

KDE + 1-hidden layer neural net

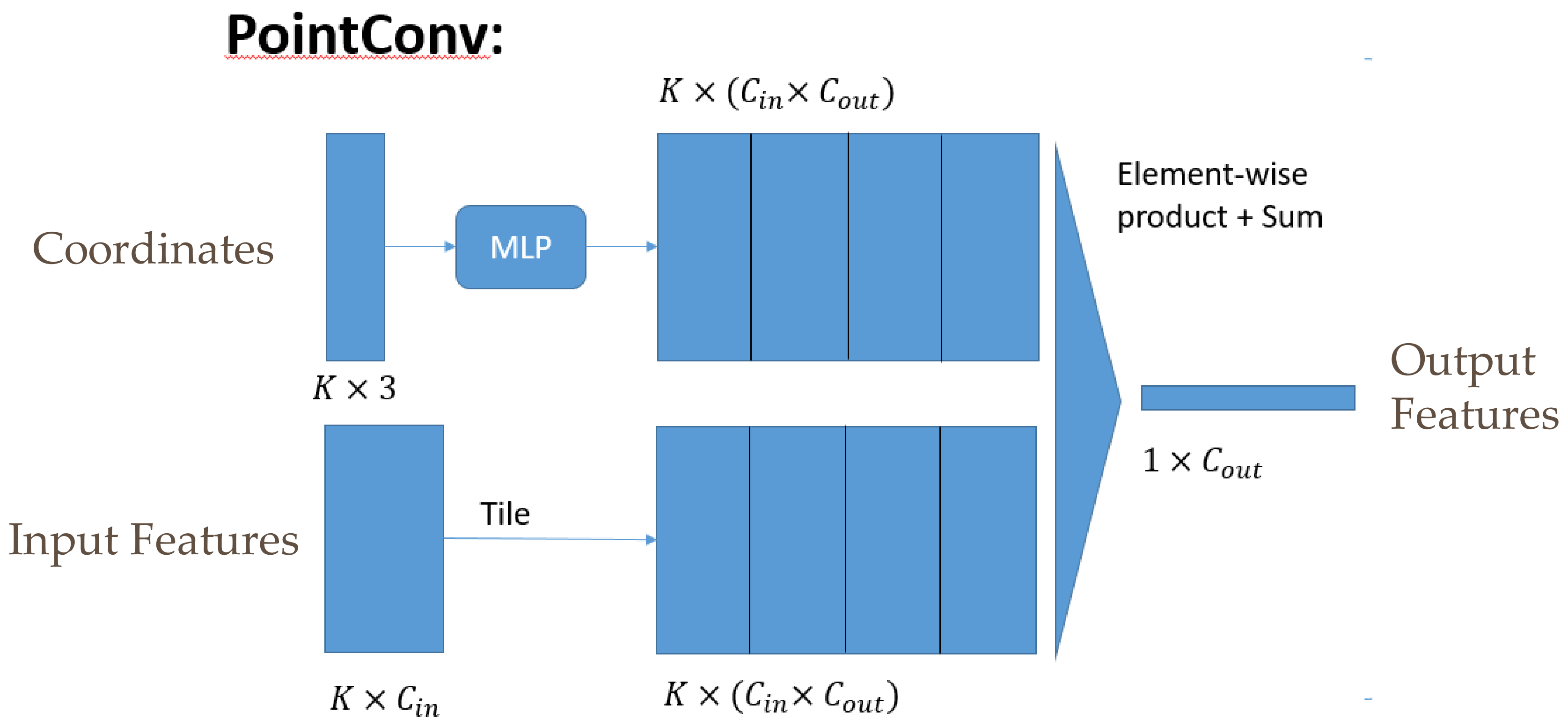


(a) (b) (c) (d)

W. Wu, Z. Qi, **FL**, arXiv:1811.07246

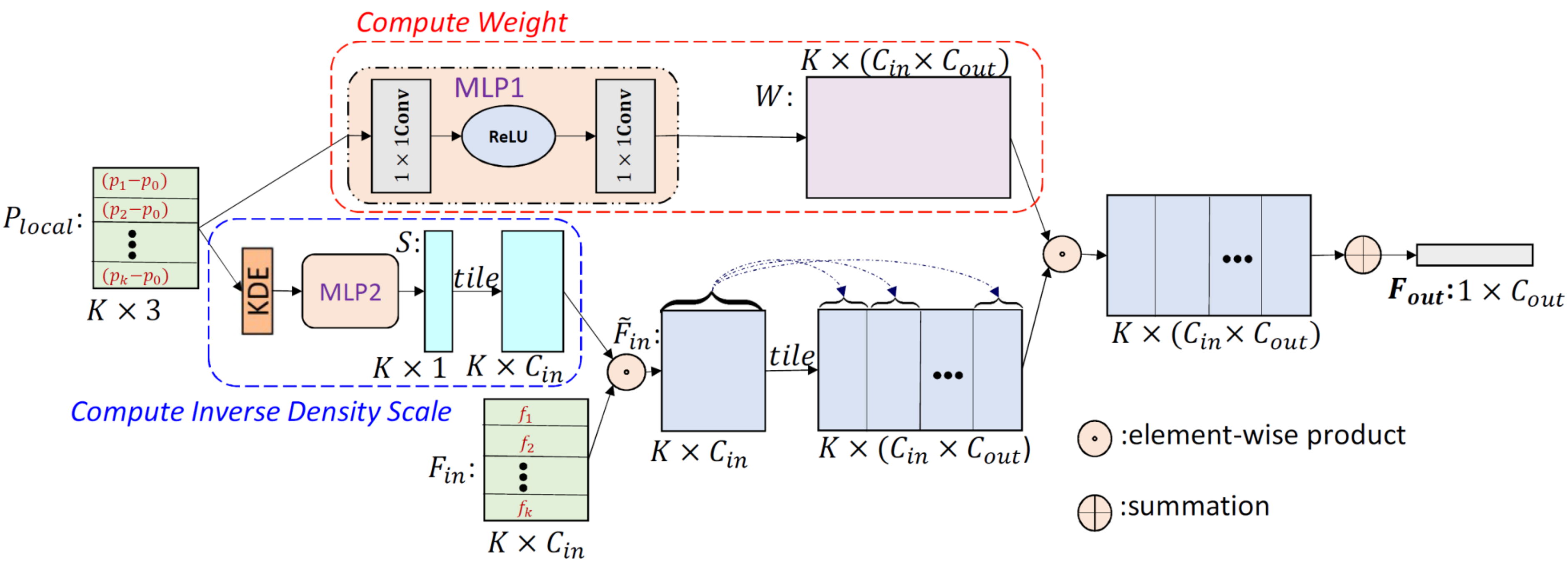
PointConv Weight Architecture

- Work on k nearest neighbors of each point



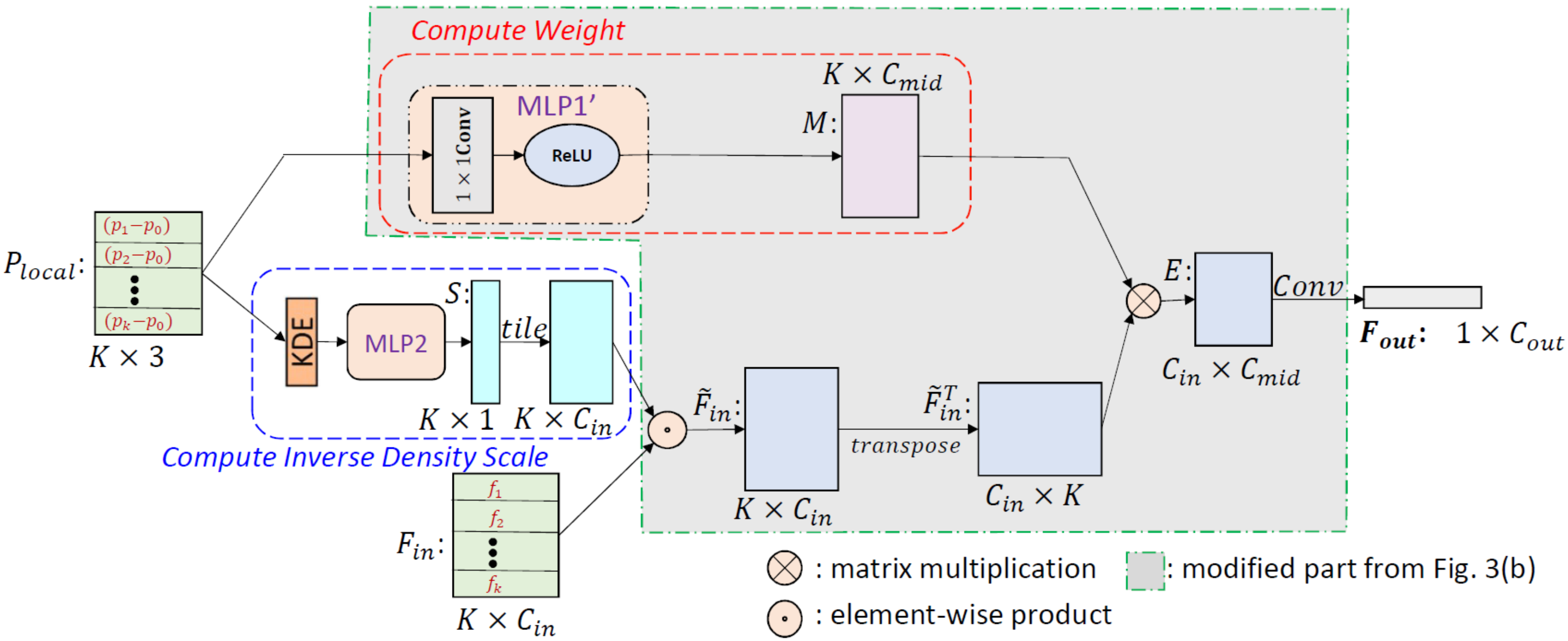
W. Wu, Z. Qi, **FL**, arXiv:1811.07246

PointConv Entire Architecture



W. Wu, Z. Qi, **FL**, arXiv:1811.07246

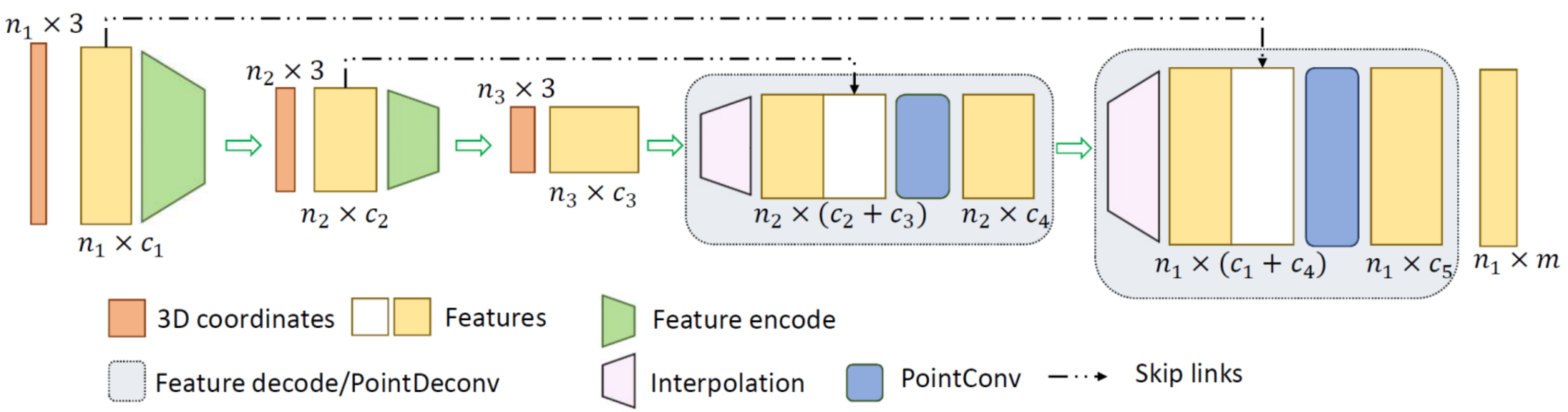
Efficient Computation



W. Wu, Z. Qi, **FL**, arXiv:1811.07246

Convolution and Deconvolution

- Both downsampling and upsampling are easy for point clouds
- Hence both convolution and deconvolution can be done
- Easily mimic a U-Net architecture for segmentation with PointConv, downsampling and upsampling layers



W. Wu, Z. Qi, **FL**, arXiv:1811.07246

PointConv Results

- 5-layer PointConv matching 7-layer AlexNet on CIFAR-10
- Proving that PointConv is real convolution

	Accuracy
SpiderCNN (Xu et al. 2018)	84.07
PointCNN (Li et al. 2018)	80.22
Image Convolution	**88.52**
PointConv 5-layer	**89.13**
AlexNet (Krizhevsky et al. 2012)	**89.00**

PointConv:25,64-1024
max_pool
PointConv:25,192-256
max_pool
PointConv:9,384-64
PointConv:9,384-64
PointConv:9,256-64
max_pool
fc:192
fc:64

W. Wu, Z. Qi, **FL**, arXiv:1811.07246

PointConv Results on ModelNet, ShapeNet and ScanNet

CAD Model: ModelNet40

Table 1. **ModelNet40 Classification Accuracy**

Method	Input	Accuracy(%)
Subvolume [25]	voxels	89.2
ECC [31]	graphs	87.4
Kd-Network [16]	1024 points	91.8
PointNet [24]	1024 points	89.2
PointNet++ [26]	1024 points	90.2
PointNet++ [26]	5000 points+normal	91.9
SpiderCNN [41]	1024 points+normal	92.4
PointConv	1024 points+normal	**92.5**

CAD Model: ShapeNet

Table 2. **Results on ShapeNet part dataset.** Class avg. is the mean IoU averaged across all object categories, and inctance avg. is the mean IoU across all objects.

	class avg.	instance avg.
SSCNN [42]	82.0	84.7
Kd-net [16]	77.4	82.3
PointNet [24]	80.4	83.7
PointNet++[26]	81.9	85.1
SpiderCNN [41]	82.4	85.3
SPLATNet$_{3D}$ [33]	82.0	84.6
PointConv	**82.8**	**85.7**

Realistic Indoor Data: ScanNet

Table 3. **Semantic Scene Segmentation results on ScanNet**

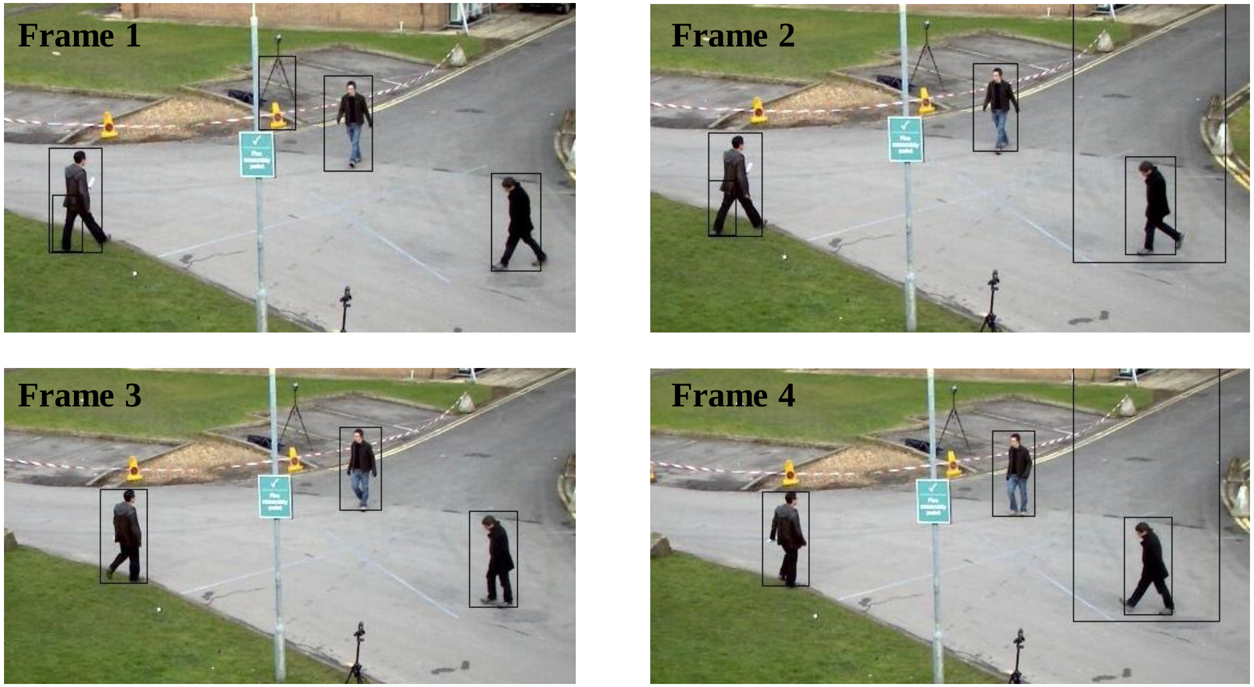
Method	mIoU(%)
ScanNet [5]	30.6
PointNet++ [26]	33.9
SPLAT Net [33]	39.3
Tangent Convolutions [35]	43.8
PointConv	**55.6**

Today’s Talk

- Understanding and New Designs on CNNs
- **Multi-Target Tracking with bilinear LSTM**
  - Novel LSTM model coming from studies on tracking

C. Kim, **FL**, J. Rehg. ECCV 2018

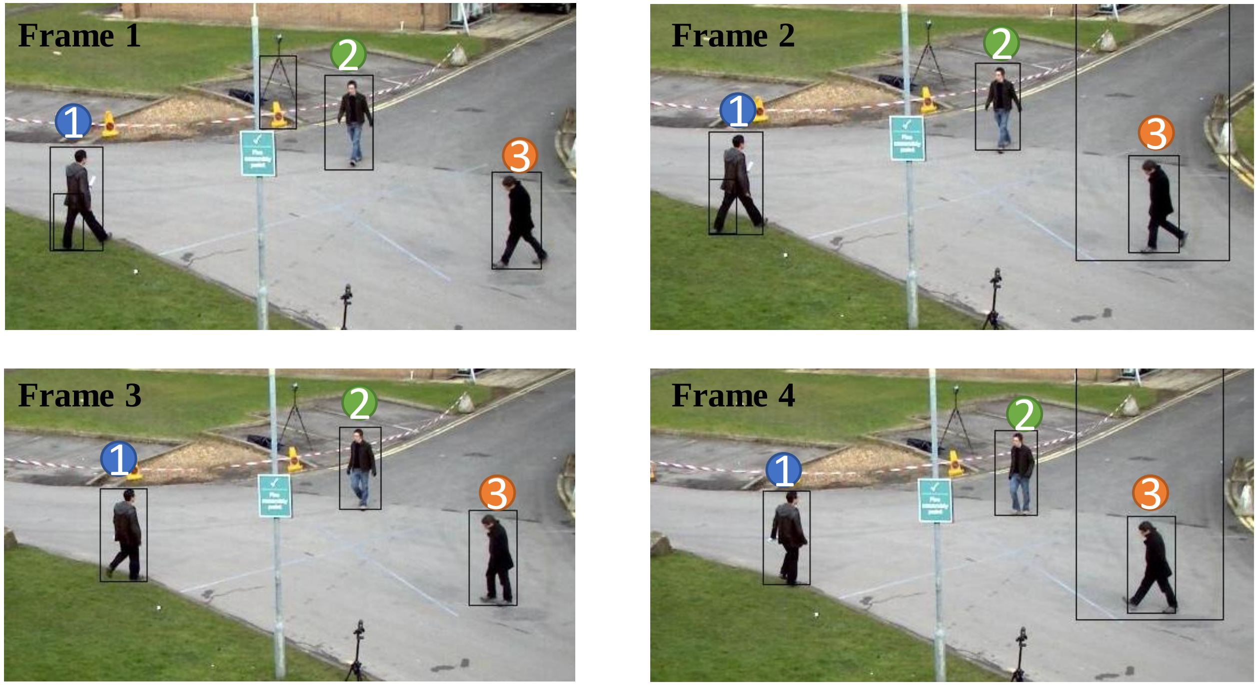
Multi-Target Tracking by Detection



Link person detections in each frame into tracks

Search space reduced by using a person detector

Multi-Target Tracking by Detection



Link person detections in each frame into tracks

Search space reduced by using a person detector

Multi-Target Tracking Illustration

The Essence of Tracking

Appearance Cues

- People (targets) look different, they wear different clothes

Motion Cues

- People (targets) move in a smooth/piecewise-smooth manner

Appearance Cues

Identity (ID) Switch!

Multiple Appearances + Motion

Successful tracking algorithms combine appearance and motion cues

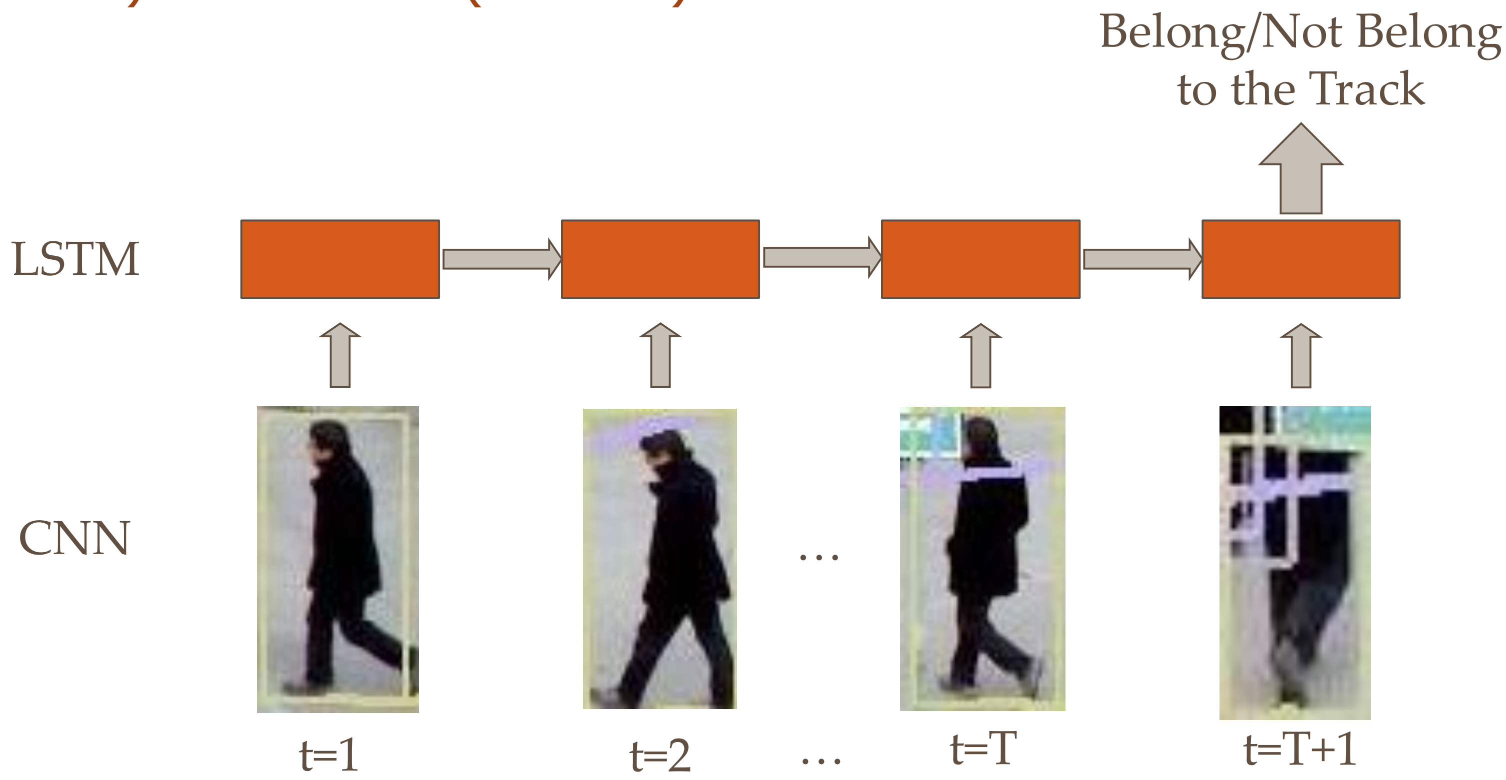
Each object can have many appearances, this needs to be handled too

Goal: End-to-End Training

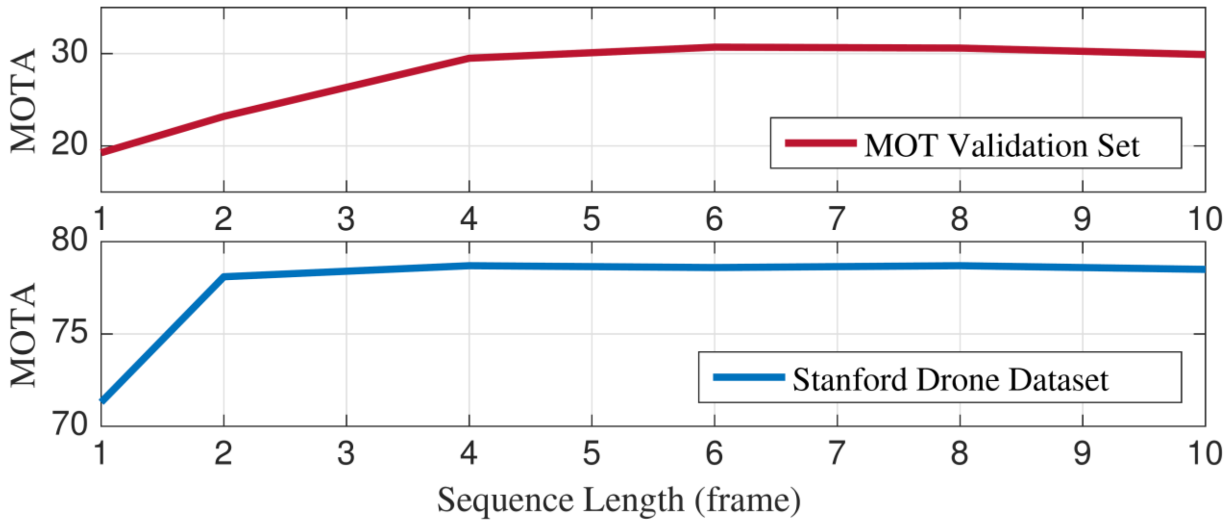
- Interestingly, tracking is rarely trained end-to-end
  - There is often an appearance model that is updated online
    - e.g. MHT-DAM [Kim et al. 2015], STAM [Chu et al. 2017]
  - And then a motion model that is separately updated
    - Most likely, a heuristic motion model (linear, constant velocity)
    - Or Kalman filter (e.g. [Kim et al. 2015])
  - And then post-processing
- There should be a few benefits for end-to-end training
  - Using more complex nonlinear motion models
  - Have the motion and appearance models better work together

Previous attempts on using a recurrent model

- A standard approach to train on a video sequence would be a convolution + recurrent model
  - Tried a couple of times (Milan et al. 2017, Sadeghian et al. 2017) with some (limited) success



Interesting Phenomenon on a Recurrent Model



(b)

Using longer sequences to train the LSTM does not seem to bring any benefit!

(image cf. Sadeghian et al. 2017)

Reflect about this Longer Training Sequence issue:

Appearance Part

Multiple Appearances!

Longer sequence in training should be beneficial

Motion Part

Single Motion Trajectory!

Longer sequence may not be beneficial

Longer Training Sequence

Appearance Part

Multiple Appearances!

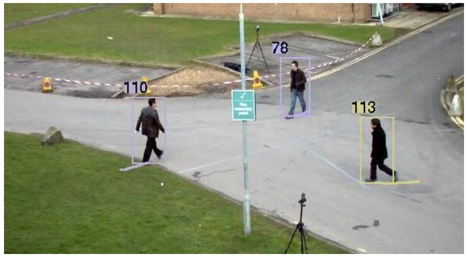
Longer sequence in training
should be beneficial

Hypothesis:

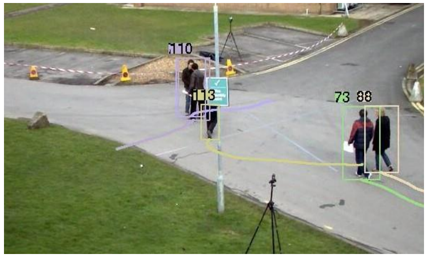
LSTM in multi-target tracking may **not** be modeling multiple appearances properly

The Dilemma of the LSTM Memory

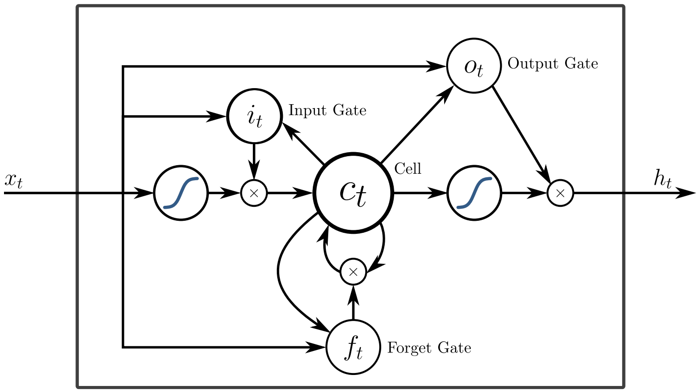
Memory



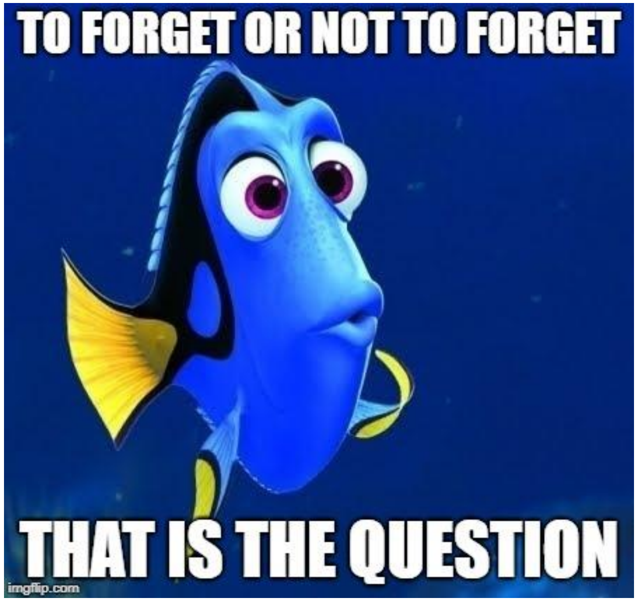
x_t



LSTM



Why is there not an option of:
put the memory aside?



In the Quest for a New LSTM

- We check a non-deep appearance modeling approach
- Recursive least squares
  - Used in several work, e.g. DCF/KCF (Henriques et al. 2012), SPT (Li et al. 2013), MHT-DAM (Kim et al. 2015)
  - As well as being a classic tracking approach in robotics
  - Global optimal online appearance modeling framework
  - Appearance model is a classifier/regressor
  - Capable of modeling multiple appearances

How does it work

- Tracker as a regressor
  - Appearance model: classifies any new appearance to object/not object

$$w_t = \arg\min_{w} ||w^\top x_{0:t} - y_{0:t}||^2 + \lambda ||w||^2$$

Appearance Features (e.g. CNN) from Positive and Negative Examples

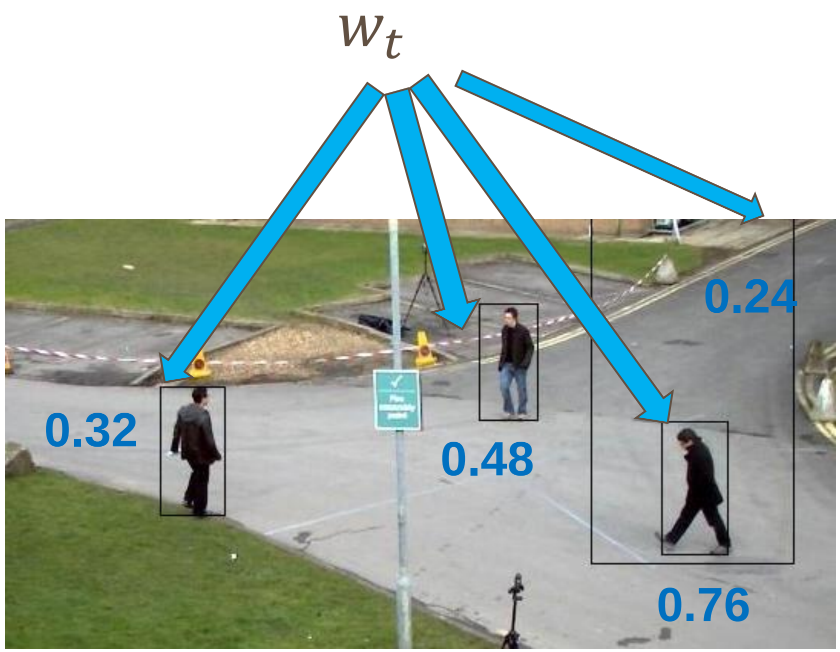
(Soft) Labels
e.g. Jaccard index

Negative (label = 0)

Positive (label = 1)

Testing and recursive training

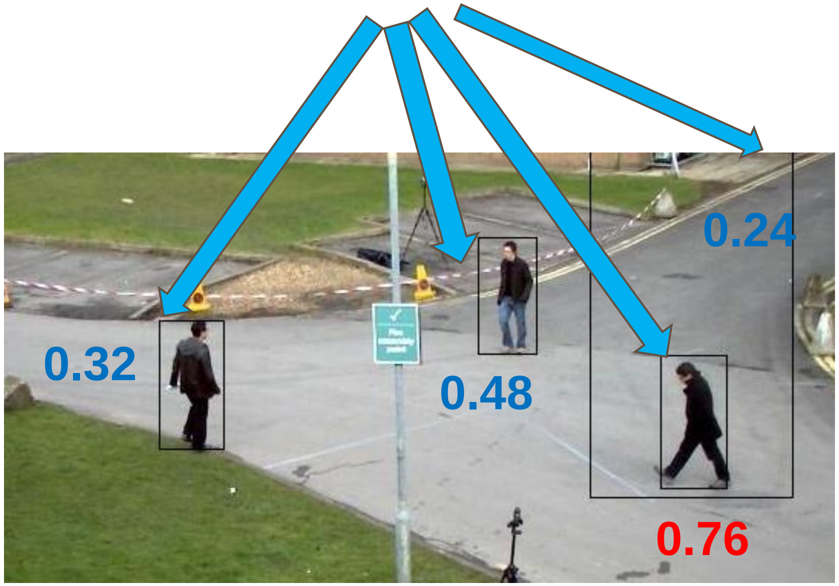
- Test model on all detections:



Testing and recursive training

- Decide which one is matched to the track

w_t



Testing and recursive training

- Generate training examples for time t+1
- Solve for w_{t+1}

$$w_{t+1} = \arg\min_{w} ||w^{\top} x_{0:t+1} - y_{0:t+1}||^2 + \lambda ||w||^2$$

(Some of the) good stuff with least squares

Solution of w:

$$\boldsymbol{w} = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y} = (\boldsymbol{H} + \lambda \boldsymbol{I})^{-1} \boldsymbol{c}$$

$$\boldsymbol{H}_k = \boldsymbol{X}_{(1:k-1)}^\top \boldsymbol{X}_{(1:k-1)} + \mathbf{X}_{(k)}^\top \mathbf{X}_{(k)}$$

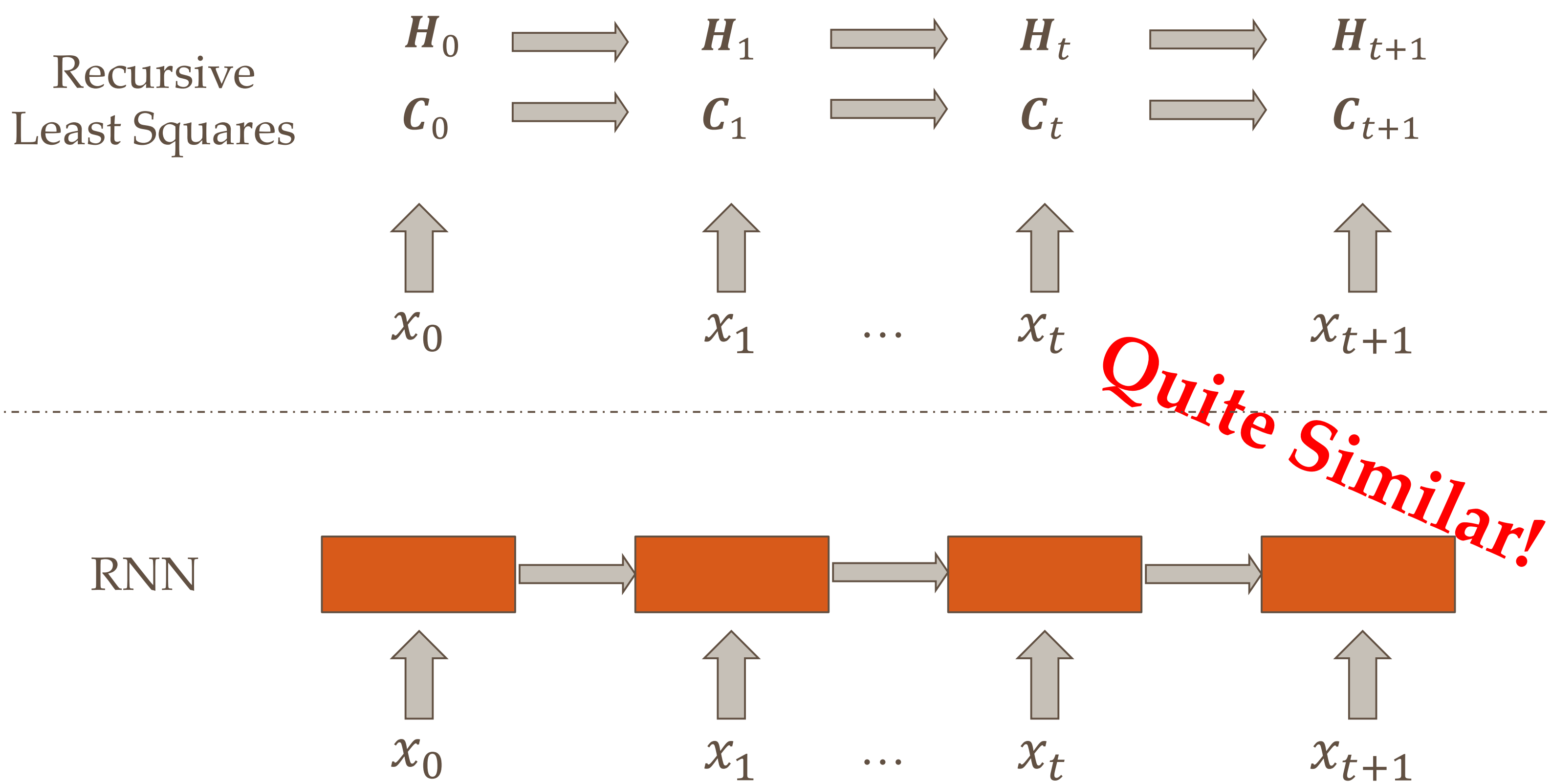
$$\boldsymbol{c}_k = \mathbf{X}_{(1:k-1)}^\top \mathbf{y}_{(1:k-1)} + \mathbf{X}_{(k)}^\top \mathbf{y}_{(k)}$$

1) Each frame is separable!
2) Inversion does not depend on number of targets (tracks)

- In DCF/KCF (Henriquez et al. 2012, 2014), more computational savings with Fourier domain transformations
- In MHT-DAM (Kim et al. 2015), this is used to learn a different appearance model for each branch in an MHT

The “Recurrent Model” Version of Least Squares

Problem: Storing $d \times d$ matrix H in RNN is too memory-consuming



Low-rank Approximation

- Go back to the solution formula

$$\boldsymbol{w} = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y} = (\boldsymbol{H} + \boldsymbol{\lambda} \boldsymbol{I})^{-1} \boldsymbol{c}$$

$$\boldsymbol{w}^\top \boldsymbol{x} \approx \sum_{i=1}^{r} \boldsymbol{c}^\top \boldsymbol{h}_i \boldsymbol{h}_i^\top \boldsymbol{x} = \sum_{i=1}^{r} \mu_i \boldsymbol{h}_i^\top \boldsymbol{x}$$

Feature input (e.g. CNN)

Track-specific layer

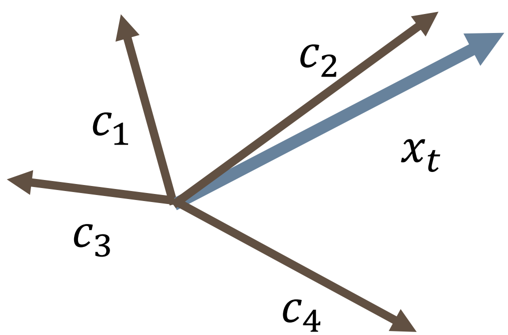
Memory

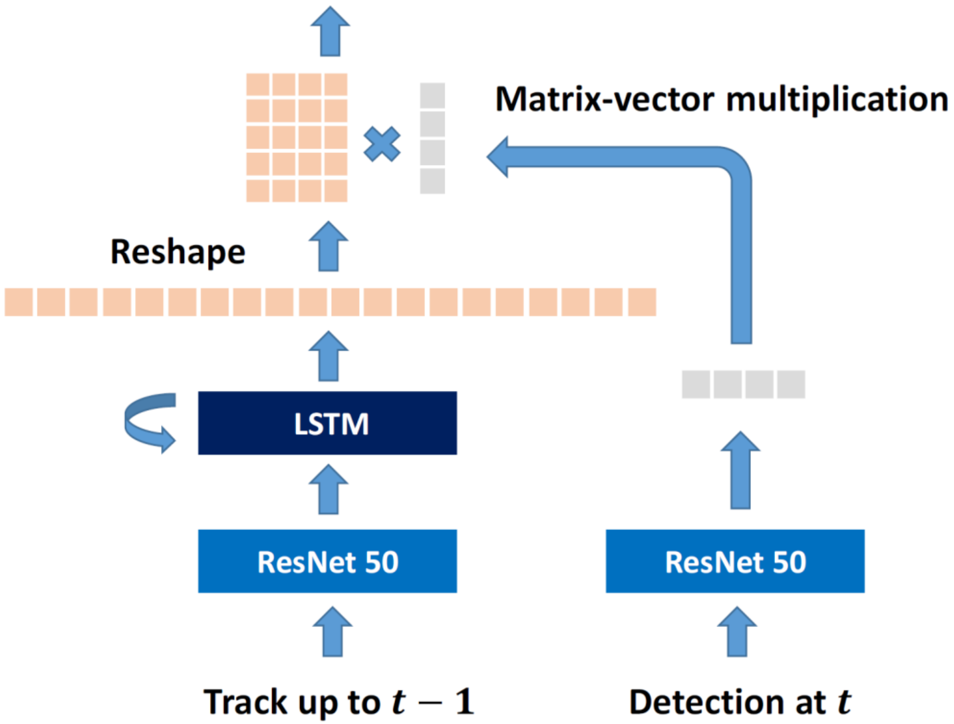
The difference between this and a normal RNN/LSTM update?

Bilinear LSTM

Adopt multi-modality in LSTM

Each column can be thought of as one modality

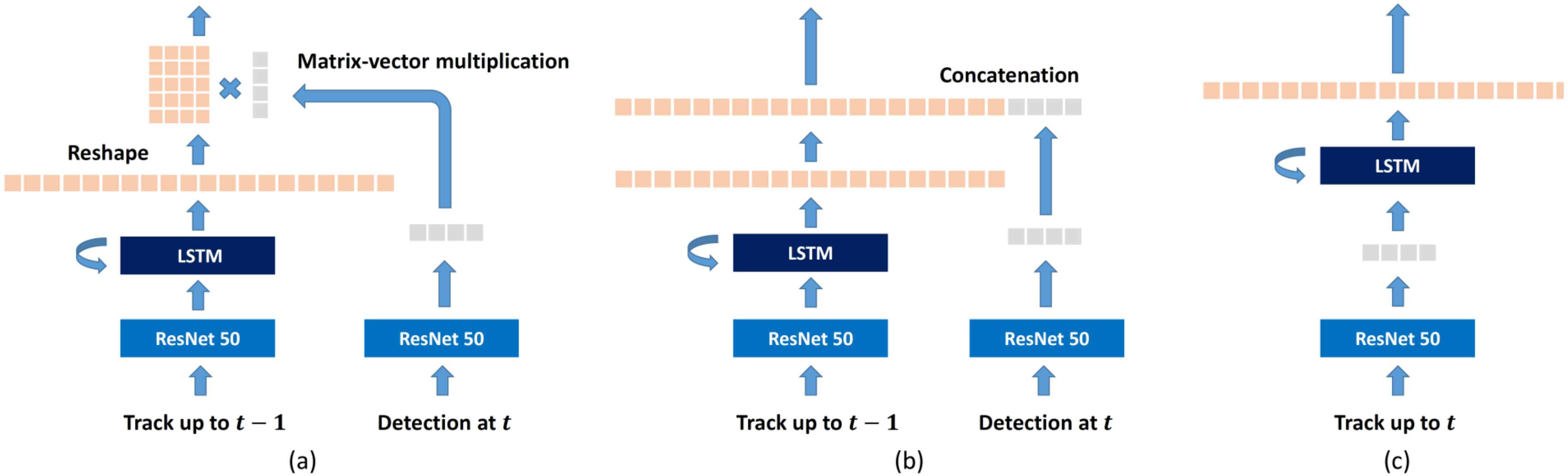




C. Kim, **FL**, J. Rehg. ECCV 2018

Bilinear LSTM Model Study

- We tried 3 models for
  - Appearance LSTM
  - Motion LSTM



Bilinear LSTM

Concatenate Memory and Input

Normal LSTM

Experiment Details

- MOT-17 dataset (without 17-09 and 17-10) + ETH + PETS + TUD + TownCentre + KITTI16 + KITTI19 as training
- MOT-17-09, MOT-17-10 as validation
- Faster R-CNN detector with ResNet 50 head
- Public Detections

Soft-max			
Matrix-vector Multiplication-relu 8			
Reshape	8×256	Reshape	256×1
LSTM	2048		
FC-relu	256	FC-relu	256
ResNet-50	2048	ResNet50	2048
Input at $t-1$	$128 \times 64 \times 3$	Input at t	$128 \times 64 \times 3$

(a)

Soft-max			
FC-relu 512			
Concatenation $2048 + 256$			
LSTM	2048		
FC-relu	256	FC-relu	256
ResNet-50	2048	ResNet50	2048
Input at $t-1$	$128 \times 64 \times 3$	Input at t	$128 \times 64 \times 3$

(b)

Soft-max	
FC-relu	512
LSTM	2048
FC-relu	256
ResNet-50	2048
Input at t	$128 \times 64 \times 3$

(c)

Comparison between different appearance LSTMs

- Bilinear LSTM significantly better than other LSTM variants
  - ID switches almost halved
- Longer training sequence make a difference

LSTM	MOTA	IDF1	IDS
Bilinear	**52.33**	**59.07**	**233**
Baseline1	50.43	51.28	412
Baseline2	50.97	51.49	462

State dim.	MOTA	IDF1	IDS
512	52.14	56.66	283
1024	52.36	55.85	**222**
2048	52.33	**59.07**	233

$N_{\max}$	MOTA	IDF1	IDS
10	51.96	54.36	271
20	52.27	58.38	228
40	52.33	**59.07**	233
80	52.32	57.21	239
160	52.41	55.19	**222**

Table 4: Ablation Study for Appearance Gating Networks. Baseline1 and Baseline2 are the networks shown in Table 2 (b) and (c) resepectively. **(Left)** State dim. = 2048, $N_{\max}$ = 40 **(Middle)** LSTM: Bilinear, $N_{\max}$ = 40, **(Right)** LSTM: Bilinear, State dim. = 2048

Comparison between different motion LSTMs

- Bilinear LSTM does not work as well as regular LSTM in motion LSTM
  - Maybe the single modality of motion LSTM makes regular LSTM more suitable

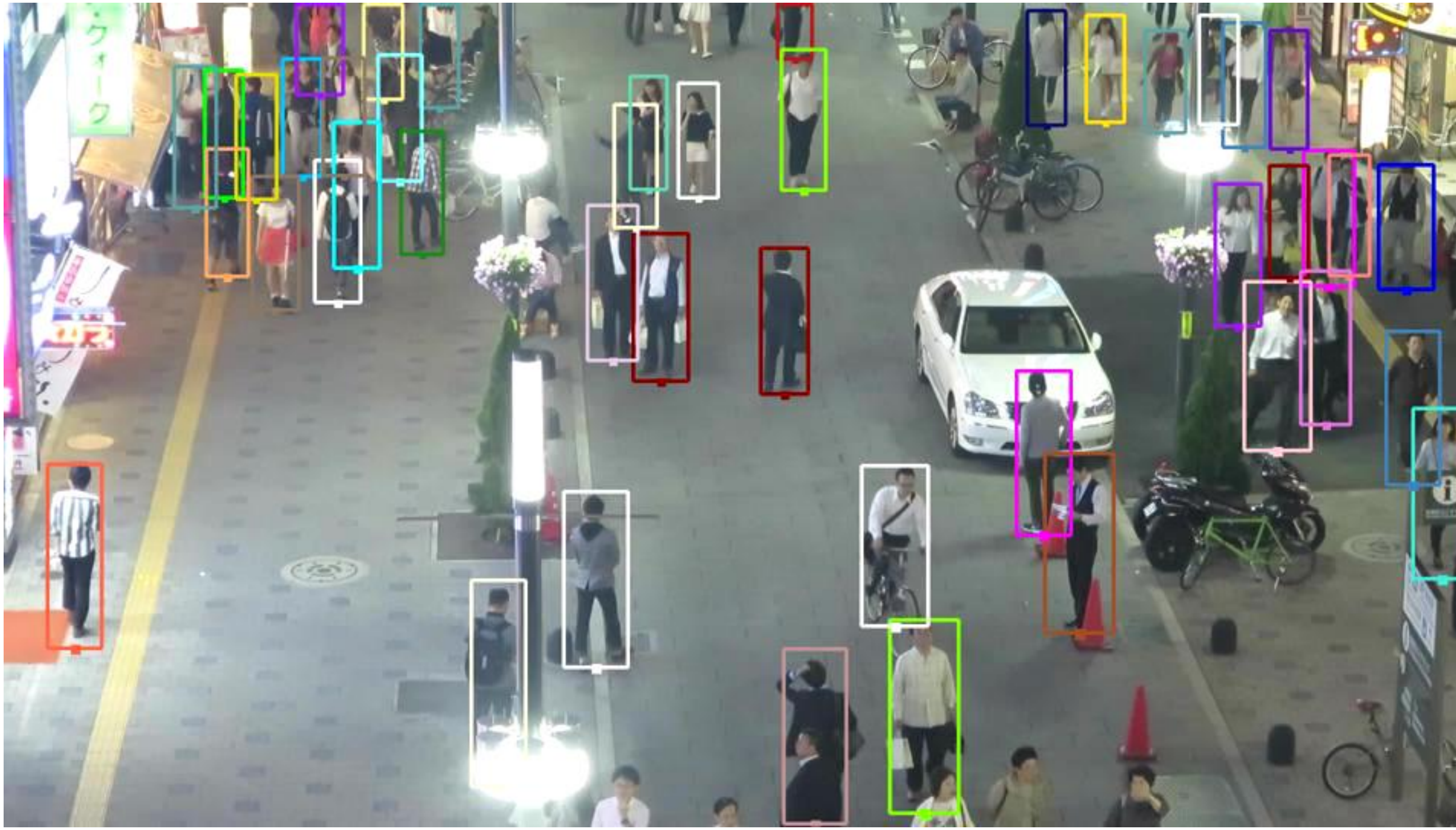
LSTM	MOTA	IDF1	IDS
Bilinear	39.68	41.22	226
Baseline1	38.90	19.38	449
Baseline2	40.14	**44.11**	**106**

State dim.	MOTA	IDF1	IDS
64	40.14	44.11	106
128	40.16	44.26	**97**
256	40.15	44.48	103

$N_{\max}$	MOTA	IDF1	IDS
20	39.76	28.50	206
40	40.14	44.11	106
80	40.15	**45.29**	104
160	40.20	**45.15**	**91**

Table 2: Ablation Study for Motion Gating Networks **(Left)** State dim. $= 64$, $N_{\max} = 40$ **(Middle)** LSTM:Baseline2, $N_{\max} = 40$, **(Right)** LSTM:Baseline2, State dim. $= 64$

Final MOT-17 Result Videos



MHT-DAM (Kim et al. 2015)

Final MOT-17 Result Videos

MHT-bLSTM

C. Kim, FL, J. Rehg. ECCV 2018

Final MOT Results

- Showing all the top non-anonymous results on MOT-17 (as of 7/31/18), sorted by IDF1:

Ours

Best in MOT 2017

Tracker	Avg Rank	MOTA	↑IDF1	MT	ML	FP	FN	ID Sw.	Frag	Hz	Detector
eHAF17 2.	13.5	51.8 ±13.2	54.7	23.4%	37.9%	33,212	236,772	1,834 (31.6)	2,739 (47.2)	0.7	Public
											TCSVT-02141-2018
jCC 3.	14.6	51.2 ±14.5	54.5	20.9%	37.0%	25,937	247,822	1,802 (32.1)	2,984 (53.2)	1.8	Public
											M. Keuper, S. Tang, Y. Zhongjie, B. Andres, T. Brox, B. Schiele. A multi-cut formulation for joint segmentation and tracking of multiple objects. In arXiv preprint arXiv:1607.06317, 2016.
MOTDT17 7.	15.8	50.9 ±11.9	52.7	17.5%	35.7%	24,069	250,768	2,474 (44.5)	5,317 (95.7)	18.3	Public
											C. Long, A. Haizhou, Z. Zijie, S. Chong. Real-time Multiple People Tracking with Deeply Learned Candidate Selection and Person Re-identification. In ICME, 2018.
MHT_bLSTM 9. NEW	20.5	47.5 ±12.6	51.9	18.2%	41.7%	25,981	268,042	2,069 (39.4)	3,124 (59.5)	1.9	Public
											C. Kim, F. Li, J. Rehg. Multi-object Tracking with Neural Gating Using Bilinear LSTM. In ECCV, 2018.
EDMT17 12.	16.4	50.0 ±13.9	51.3	21.6%	36.3%	32,279	247,297	2,264 (40.3)	3,260 (58.0)	0.6	Public
											J. Chen, H. Sheng, Y. Zhang, Z. Xiong. Enhancing Detection Model for Multiple Hypothesis Tracking. In BMTT-PETS CVPRw, 2017.
PHD_GSDL17 17.	22.8	48.0 ±13.6	49.6	17.1%	35.6%	23,199	265,954	3,998 (75.6)	8,886 (168.1)	6.7	Public
											Z. Fu, P. Feng, F. Angelini, J. Chambers, S. Naqvi. Particle PHD Filter based Multiple Human Tracking using Online Group-Structured Dictionary Learning. In IEEE Access, 2018.
FWT 26.	16.4	51.3 ±13.1	47.6	21.4%	35.2%	24,101	247,921	2,648 (47.2)	4,279 (76.3)	0.2	Public
											R. Henschel, L. Leal-Taixé, D. Cremers, B. Rosenhahn. Fusion of Head and Full-Body Detectors for Multi-Object Tracking. In Trajnet CVPRW, 2018.
MHT_DAM 28.	18.0	50.7 ±13.7	47.2	20.8%	36.9%	22,875	252,889	2,314 (41.9)	2,865 (51.9)	0.9	Public
											C. Kim, F. Li, A. Ciptadi, J. Rehg. Multiple Hypothesis Tracking Revisited. In ICCV, 2015.

C. Kim, **FL**, J. Rehg. ECCV 2018

Conclusion: Bilinear LSTM

- We proposed Bilinear LSTM as an approach to learn long-term appearance models in tracking
- Experiments show that it significantly outperforms regular LSTM, especially in terms of identity switches
  - Bilinear LSTM seems capable of learning appearance model with multiple different appearances, where traditional LSTM struggles
- We hope that this methodology can be potentially useful in other scenarios beyond tracking

Other Items

- Recruiting: I'm recruiting for students and postdocs:
- Common Sense Reasoning in Computer Vision
- Uncertainty in Machine Learning
- With application in ML Fairness

- Other research that may be interesting:
- CNN generalization theory (ICLR 2017)
- Loss function for heatmap regression in face alignment/ human pose estimation (ICCV 2019)
- Boundary flow
- CNN on underwater imaging sonar
- Open-category learning

Thank You!

Fuxin Li: http://web.engr.oregonstate.edu/~lif
Email: lif@oregonstate.edu

2077 Kelley Engineering Center,
Oregon State University
Corvallis OR 97331

I would like to thank my collaborators who contributed to the work in these slides:

Georgia Tech:
Chanho Kim, James M. Rehg

Oregon State University:
Neale Ratzlaff, Wenxuan Wu, Jialin Yuan, Zhongang Qi, Saeed Khorram, Xin Li