Efficient On-Device Models using Neural Projections

Sujith Ravi

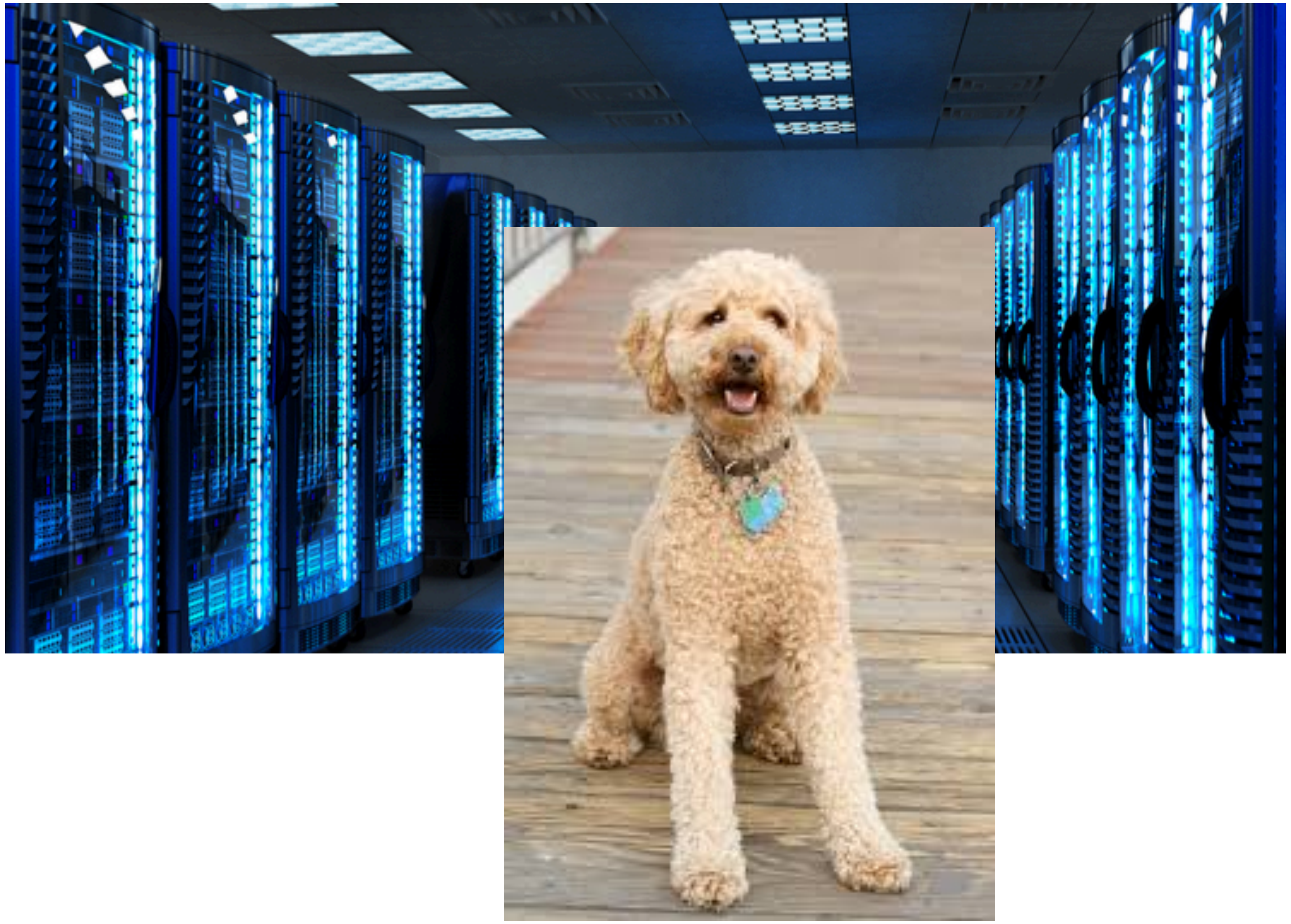
Google AI

@ravisujith

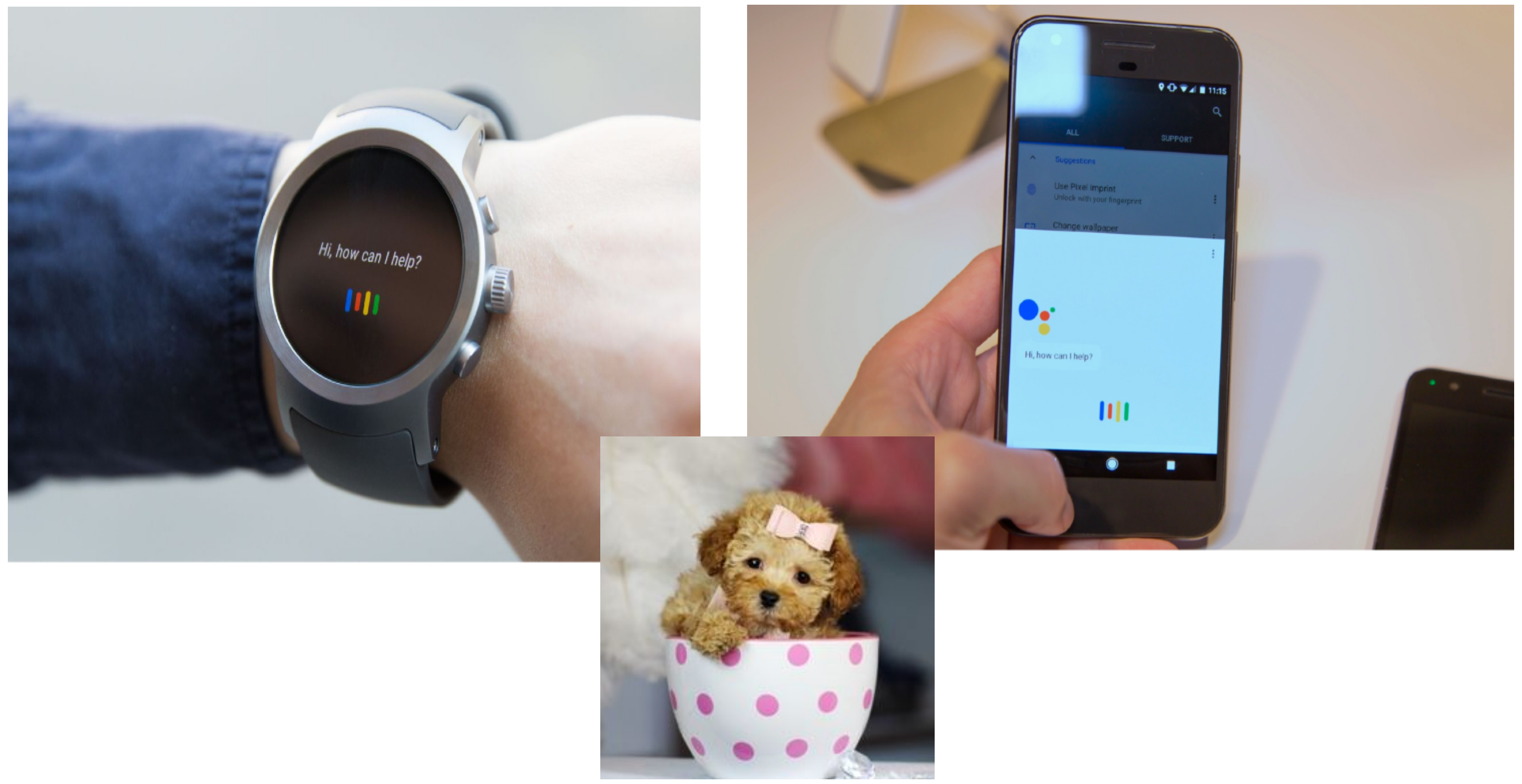
http://www.sravi.org

ICML 2019

Motivation



big Neural Networks
running **on cloud**



tiny Neural Networks
running **on device**

Sujith Ravi

User Privacy

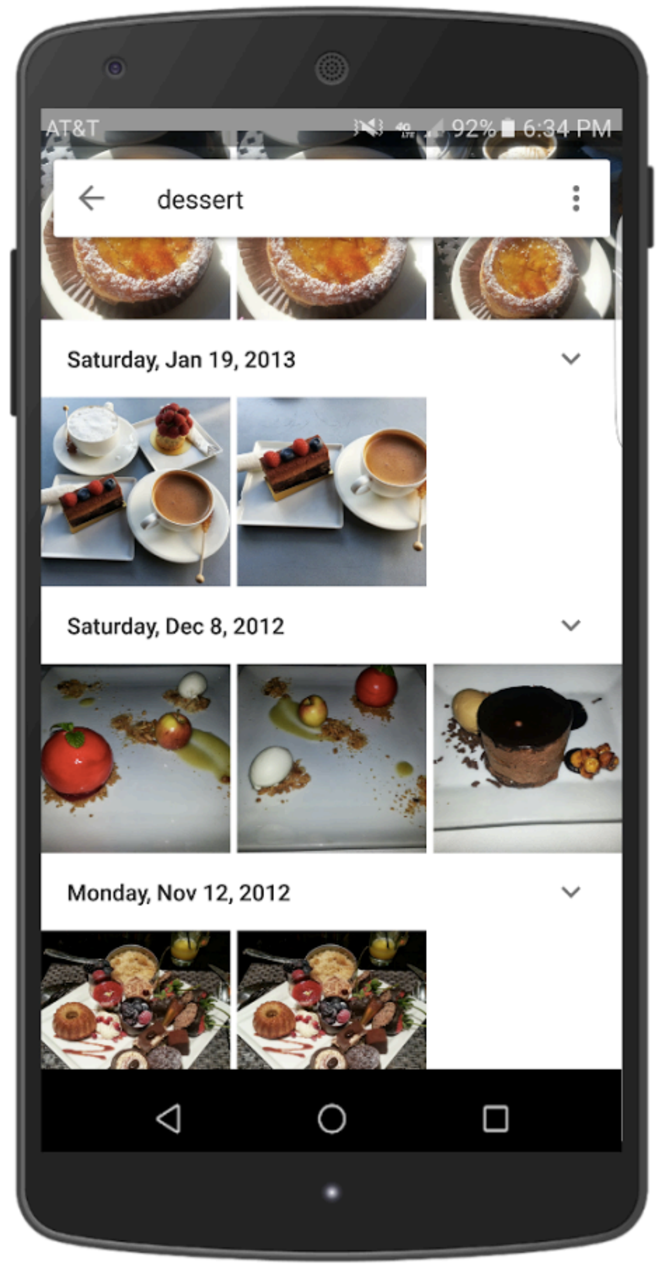
Limited Connectivity

Efficient Computing

Consistent Experience

On-Device ML in Practice

Image Recognition on your mobile phone



Smart Reply on your Android watch

Google AI Blog

“Custom On-Device ML Models with Learn2Compress”, *Sujith Ravi*
“On-Device Conversation Modeling with TensorFlow Lite”, *Sujith Ravi*
“On-Device Machine Intelligence”, *Sujith Ravi*

Challenges for Running ML on Tiny Devices

- ➡ Hardware constraints — *computation, memory, energy-efficiency*
- ➡ Robust quality — *difficult to achieve with small models*
- ➡ Complex model architectures for inference
- ➡ Inference challenging — *structured prediction, high dimensionality, large output spaces*

- Previous work, model compression
  - ➡ techniques like *dictionary encoding, feature hashing, quantization, …*
  - ➡ performance degrades with dimensionality, vocabulary size & task complexity

Can We Do Better?

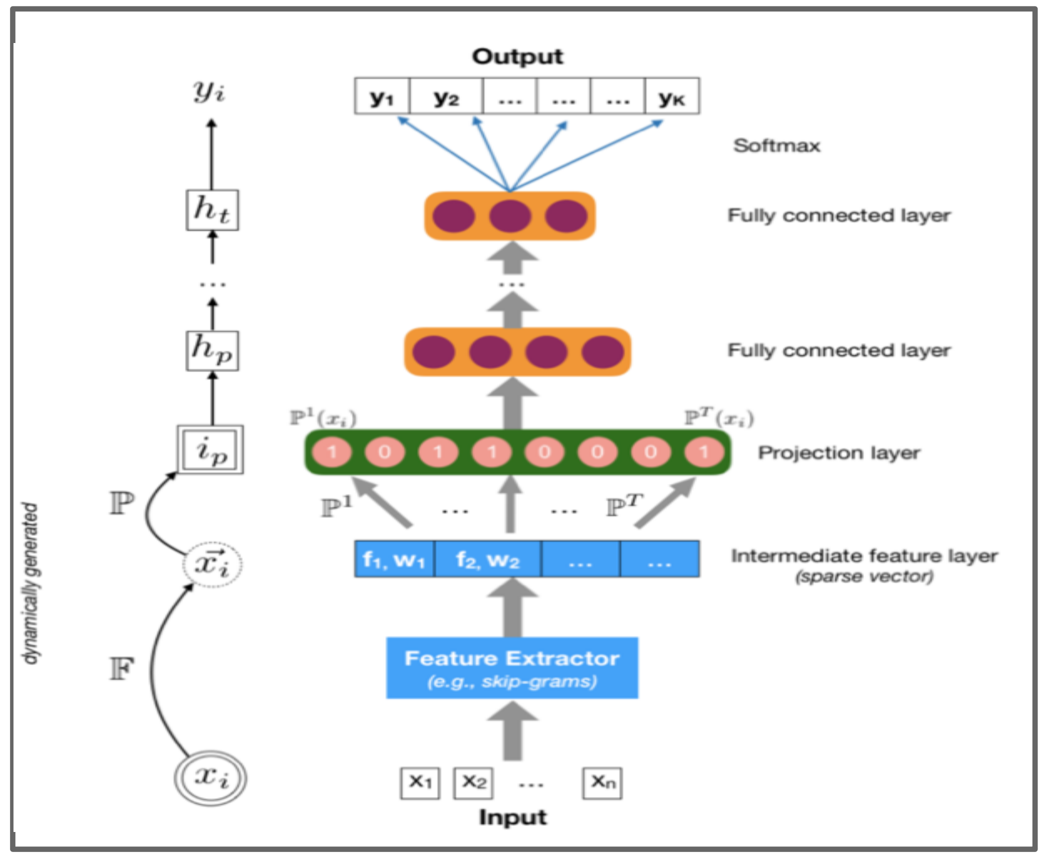
- Build on-device neural networks that
  - ➡ are small in size
  - ➡ are very efficient
  - ➡ can reach (near) state-of-the-art performance

Learn Efficient Neural Nets for On-device ML

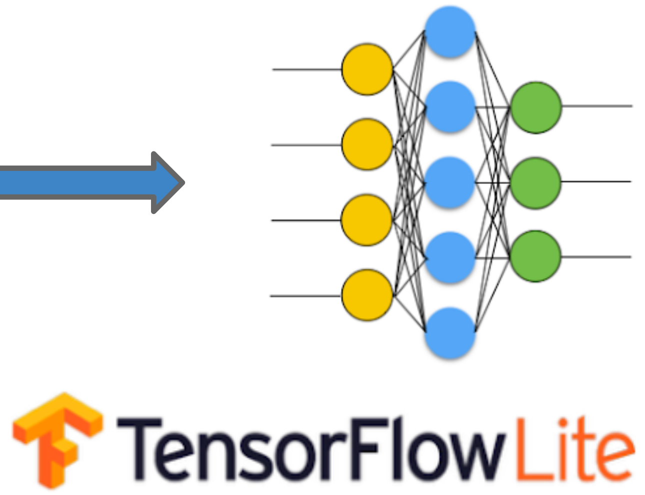
Learning
(on cloud)

Inference
(on device)

Data
(x, y)



Projection model architecture
(efficient, customizable nets)



Projection Neural Network

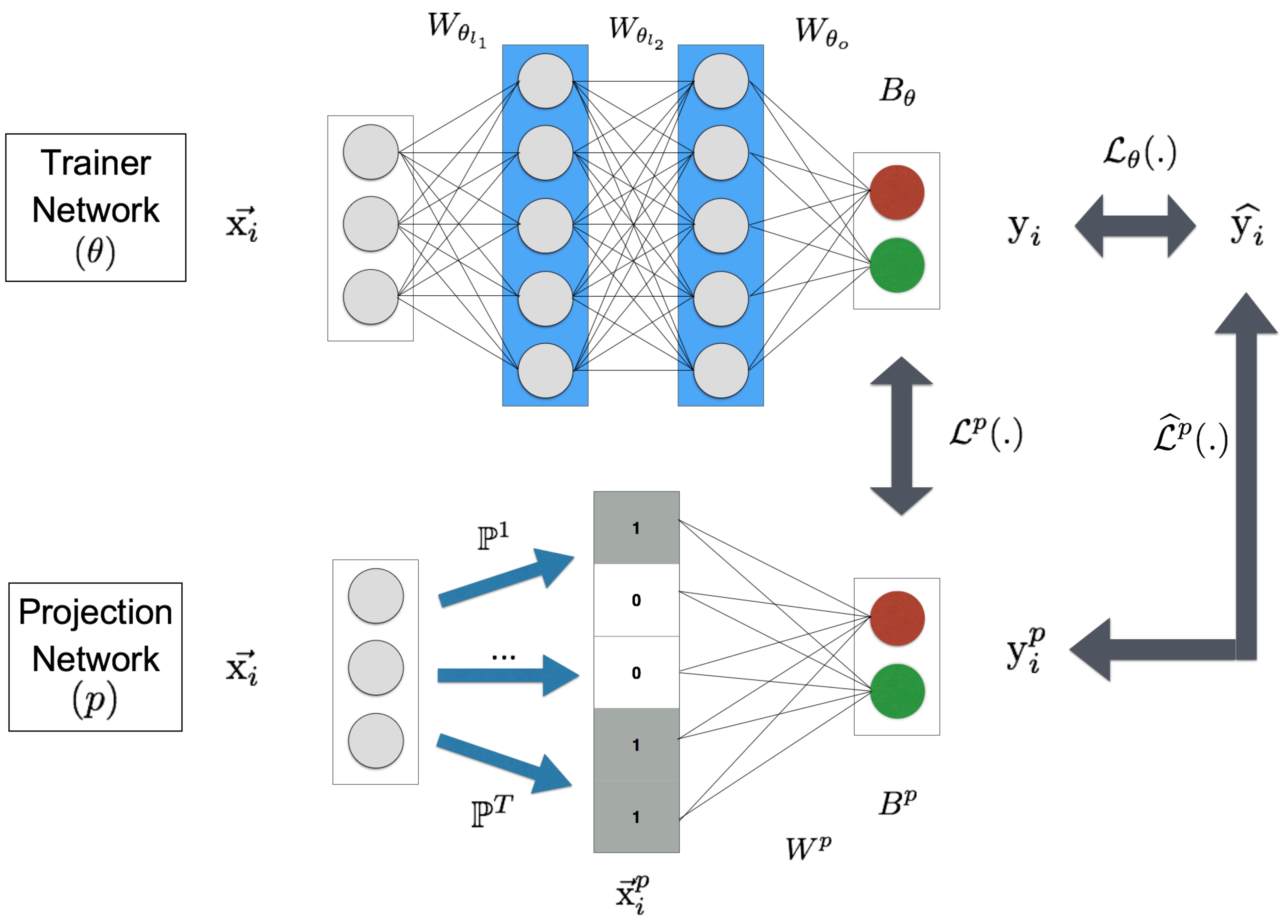
(our work) *Efficient, Generalizable Deep Networks using Neural Projections*

Optimized NN model, ready-to-use on device

- Small Size → *compact nets, multi-sized*
- Fast → *low latency*
- Fully supported inference → *TF / TFLite / custom*

Sujith Ravi

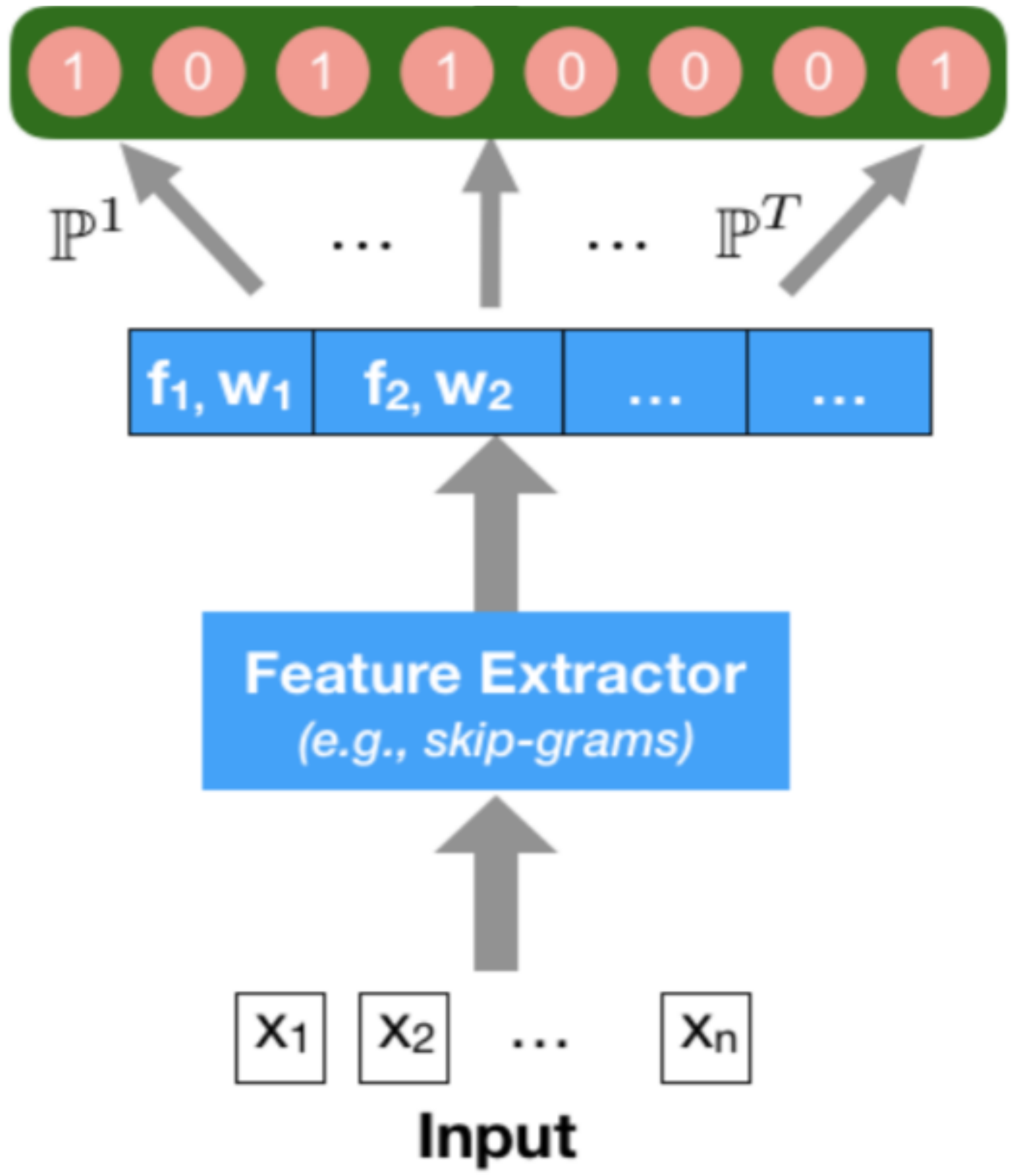
Learn Efficient On-Device Models using Neural Projections



Sujith Ravi

Projection Neural Networks

Dynamically Generated



Projection layer

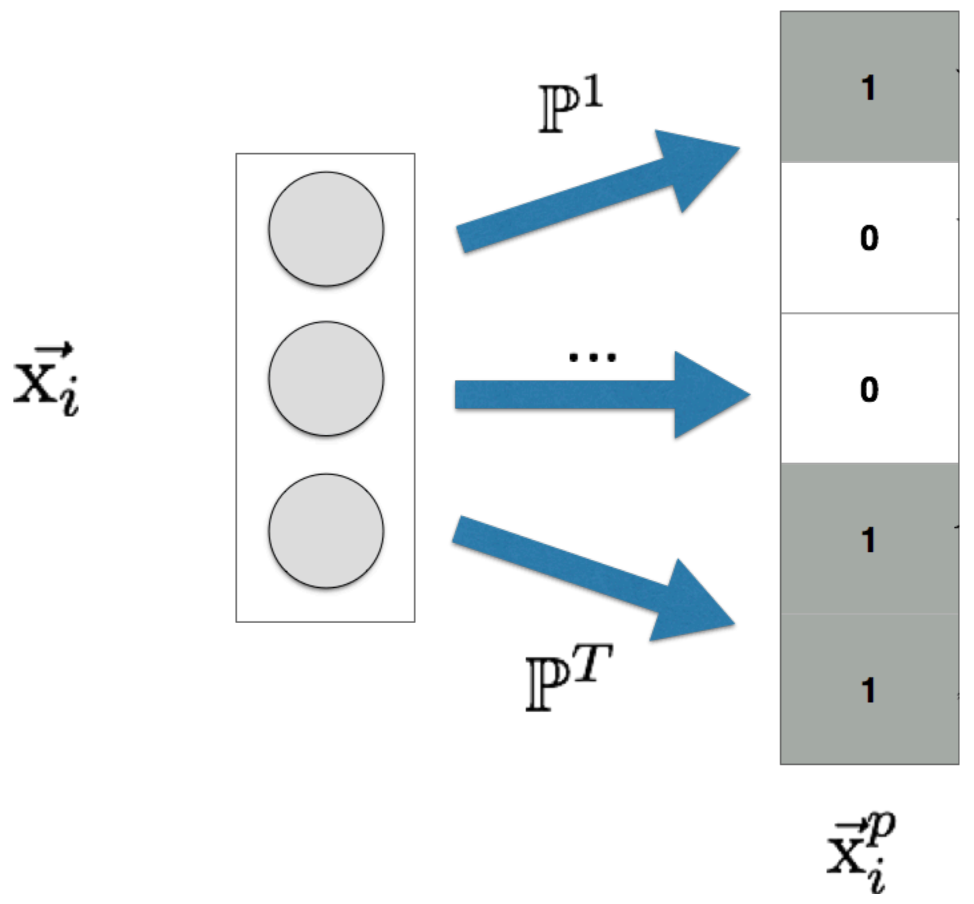
Intermediate feature layer
(sparse or dense vector)

Efficient Representations via Projections

- Transform inputs using *T* projection functions

$$\mathbb{P}^1, ..., \mathbb{P}^T$$

$$\vec{x}_i^p = \mathbb{P}^1(\vec{x}_i), ..., \mathbb{P}^T(\vec{x}_i)$$

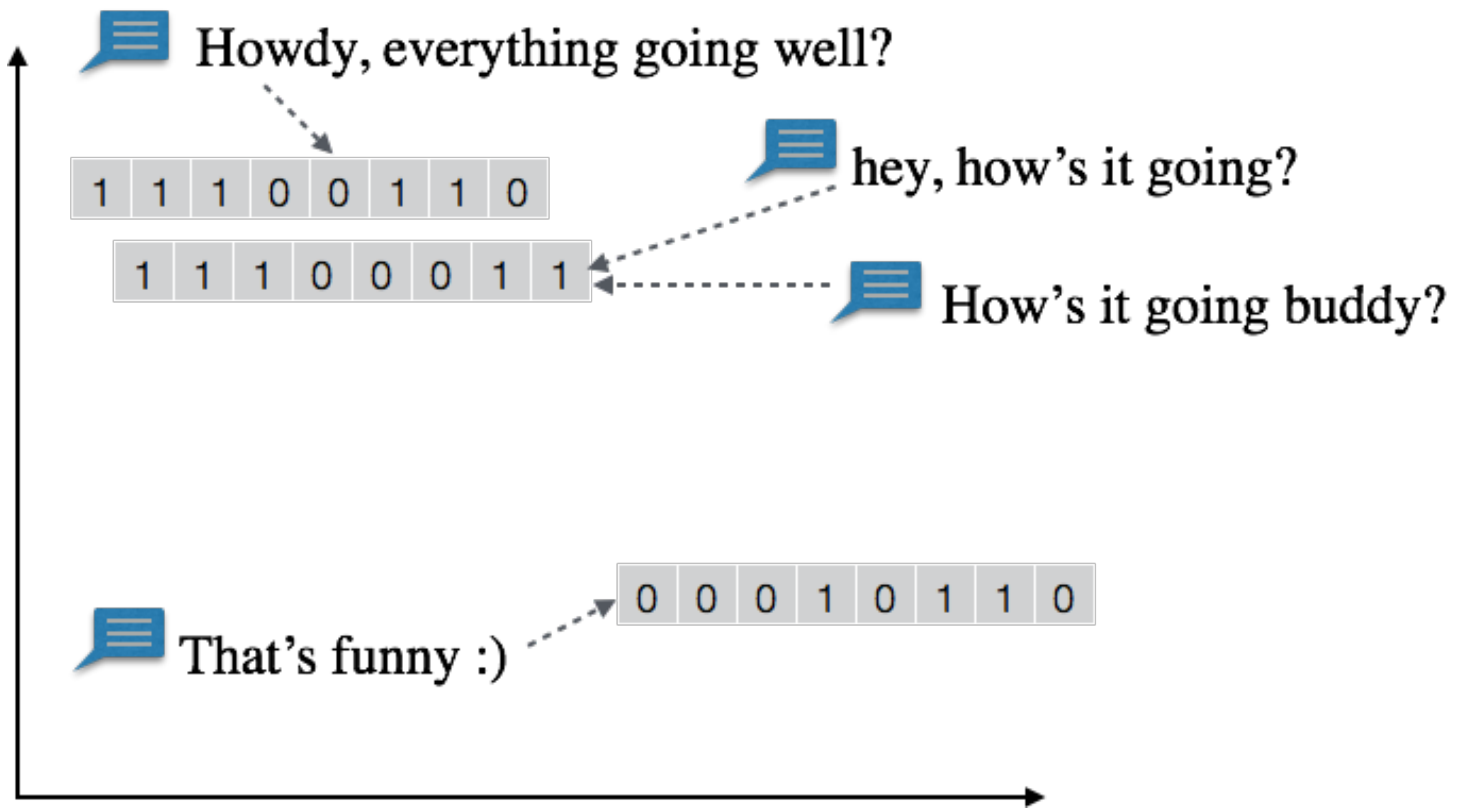


- Projection transformations (matrix) pre-computed using parameterized functions

  ➡ Compute projections efficiently using a modified version of **Locality Sensitive Hashing (LSH)**

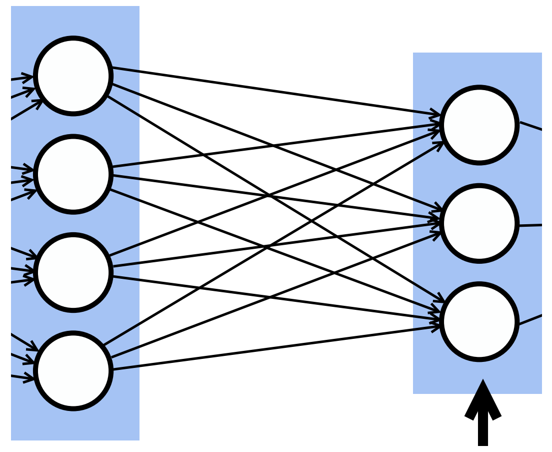
Locality Sensitive ProjectionNets

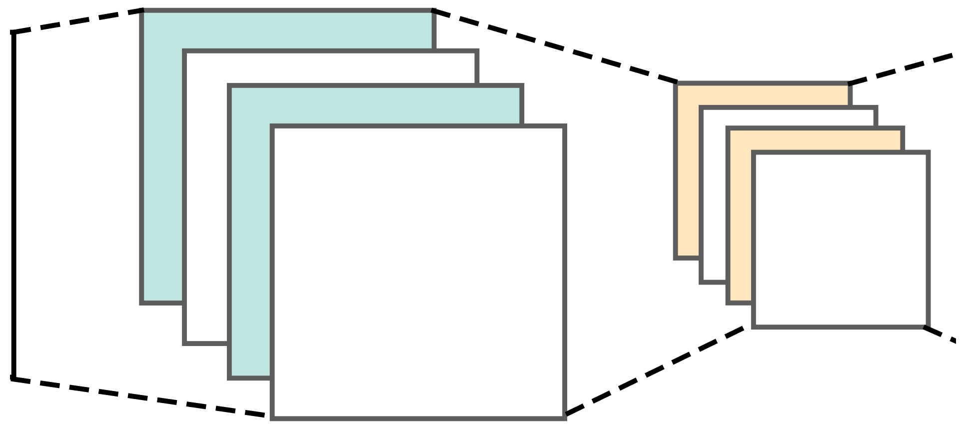
- Use randomized projections *(repeated binary hashing)* as projection operations
  - ➡ Similar inputs or intermediate network layers are grouped together and projected to nearby projection vectors
  - ➡ Projections generate compact bit (0/1) vector representations

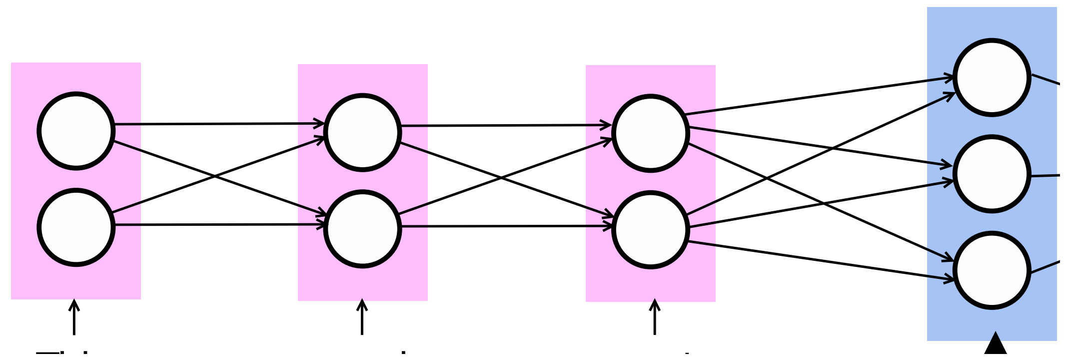


Generalizable, Projection Neural Networks

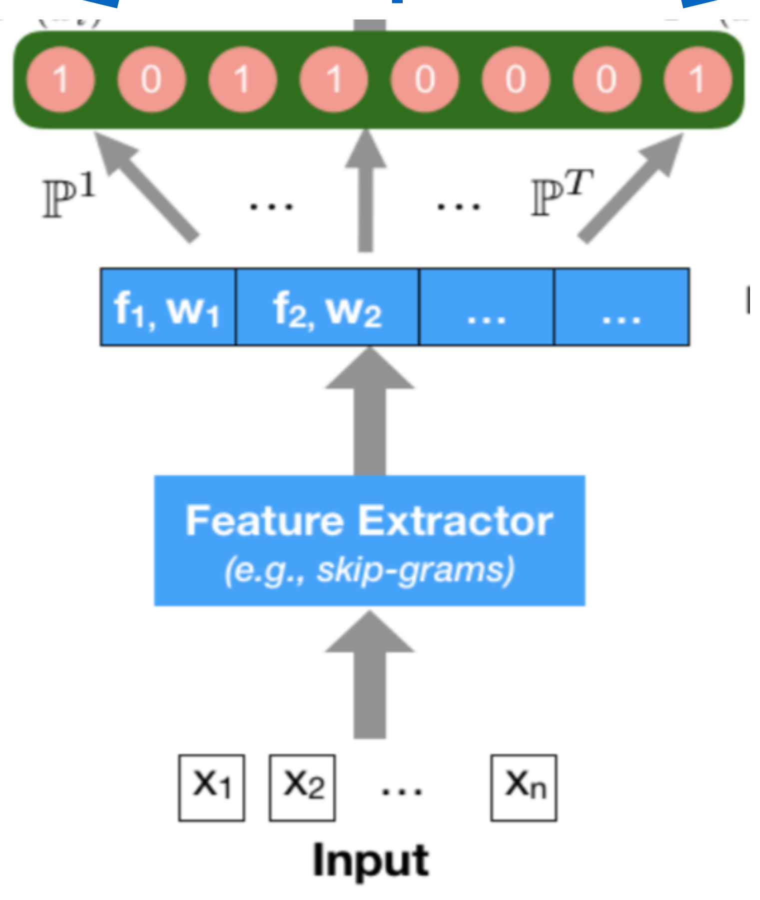
- Stack projections, combine with other operations & non-linearities to create **a family of efficient, projection deep networks**



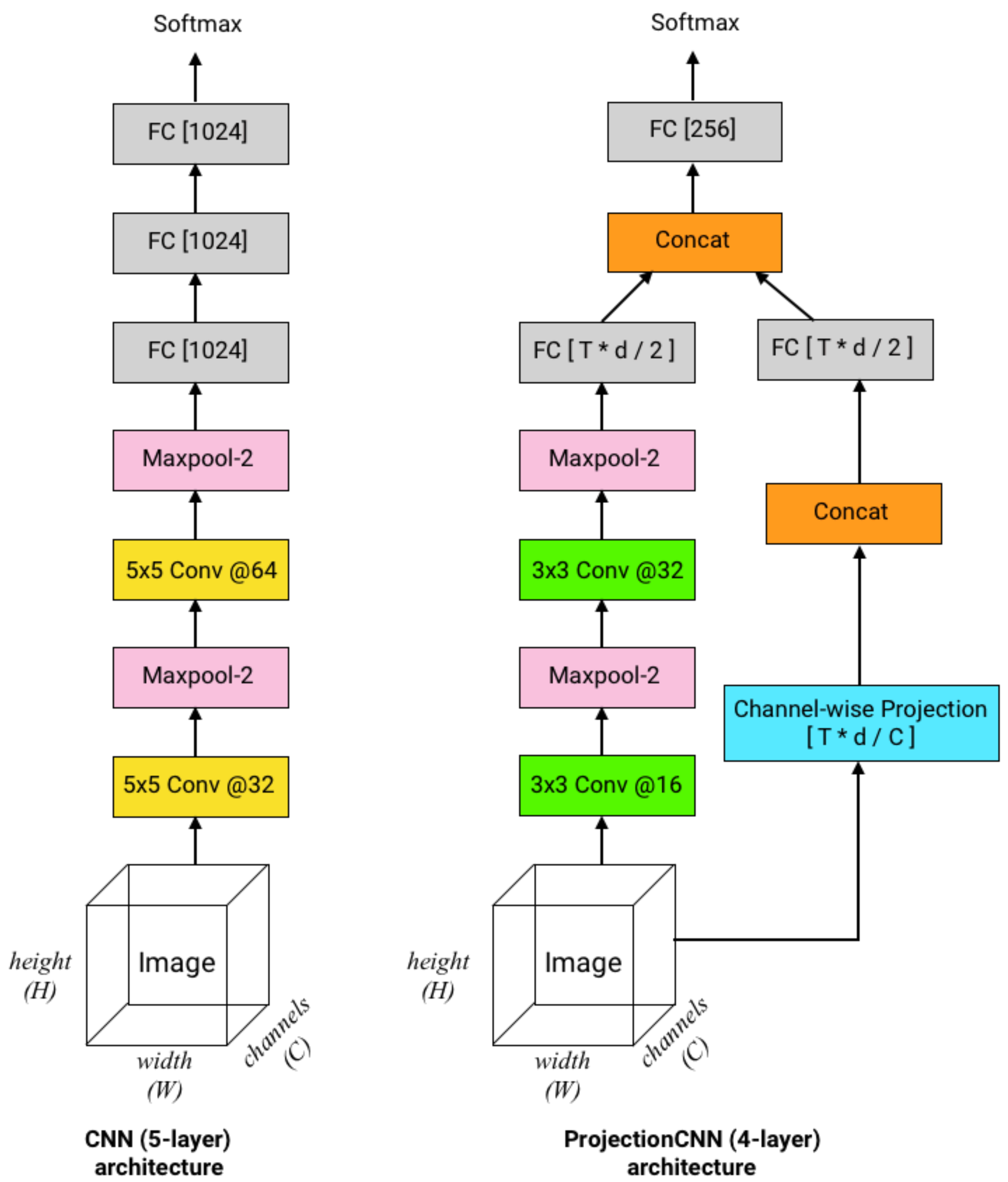




Projection + Dense | **Projection + Convolution** | **Projection + Recurrent**

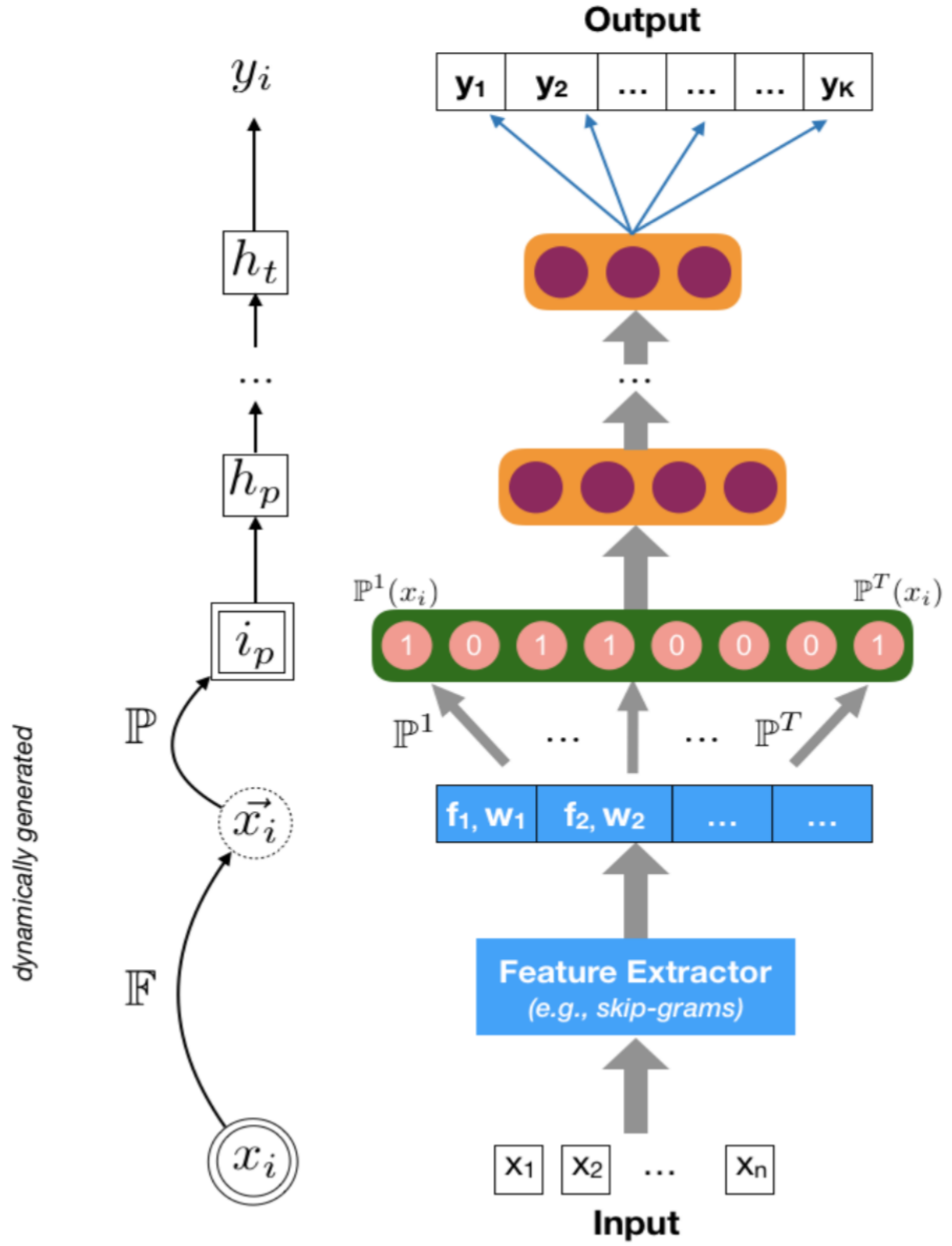


Family of Efficient Projection Neural Networks

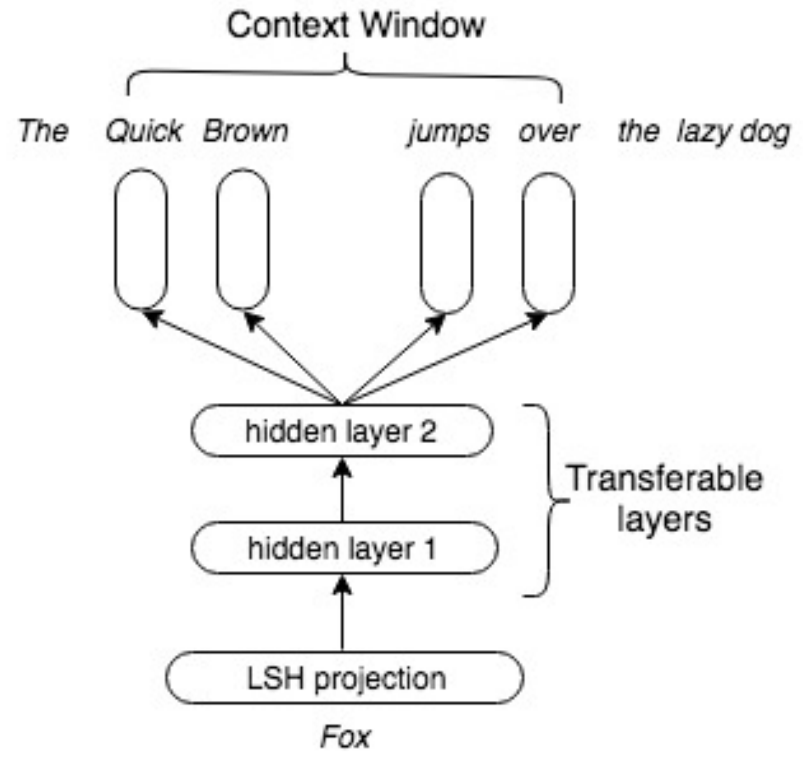


ProjectionCNN
(Ravi, ICML 2019)

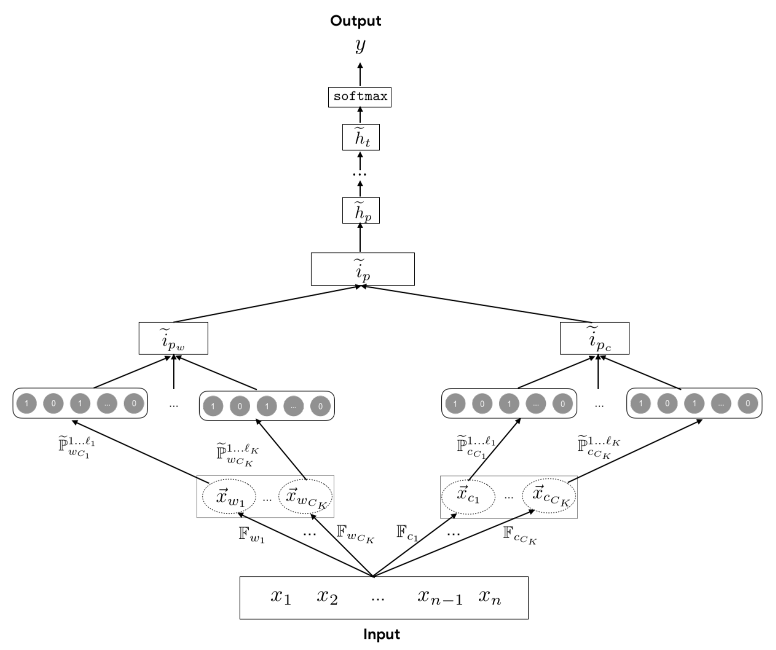
ProjectionNet
(Ravi, 2017) arxiv/abs/1708.00630



SGNN: ***Self-Governing Neural Networks***
(Ravi & Kozareva, EMNLP 2018)



Transferable Projection Networks
(Sankar, Ravi & Kozareva, NAACL 2019)



SGNN++
Hierarchical, Partitioned Projections
(Ravi & Kozareva, ACL 2019)

+… upcoming

Sujith Ravi

ProjectionNets, ProjectionCNNs for Vision Tasks

Image classification results *(precision@1)*

Model		Compression Ratio *(wrt baseline)*	MNIST	Fashion MNIST	CIFAR-10
NN (3-layer)	*(Baseline: feed-forward)*	1	98.9	89.3	-
CNN (5-layer)	*(Baseline: convolutional)* (Figure 2, Left)	0.52*	99.6	93.1	83.7
Random Edge Removal	(Ciresan et al., 2011)	8	97.8	-	-
Low Rank Decomposition	(Denil et al., 2013)	8	98.1	-	-
Compressed NN (3-layer)	(Chen et al., 2015)	8	98.3	-	-
Compressed NN (5-layer)	(Chen et al., 2015)	8	98.7	-	-
Dark Knowledge	(Hinton et al., 2015; Ba & Caruana, 2014)	-	98.3	-	-
HashNet (best)	(Chen et al., 2015)	8	98.6	-	-
NASNet-A	*(7 cells, 400k steps)* *(Zoph et al., 2018)*	-	-	-	90.5
ProjectionNet	**(our approach)** Joint *(trainer = NN)*				
	[T=8,d=10]	3453	70.6		
	[T=10,d=12]	2312	76.9		
	[T=60,d=10]	466	91.1		
	[T=60,d=12]	388	92.3		
	[T=60,d=10] + FC [128]	36	96.3		
	[T=60,d=12] + FC [256]	15	96.9		
	[T=70,d=12] + FC [256]	13	97.1	86.6	
ProjectionCNN (4-layer)	**(our approach)** (Figure 2, Right) Joint *(trainer = CNN)*	8	99.4	92.7	78.4
ProjectionCNN (6-layer)	**(our approach)** *(Conv3-64, Conv3-128, Conv3-256, ℙ [T=60, d=7], FC [128 x 256])*				
	Self *(trainer = None)*	4			82.3
	Joint *(trainer = NASNet)*	4			84.7

- Efficient wrt compute/memory while maintaining high quality

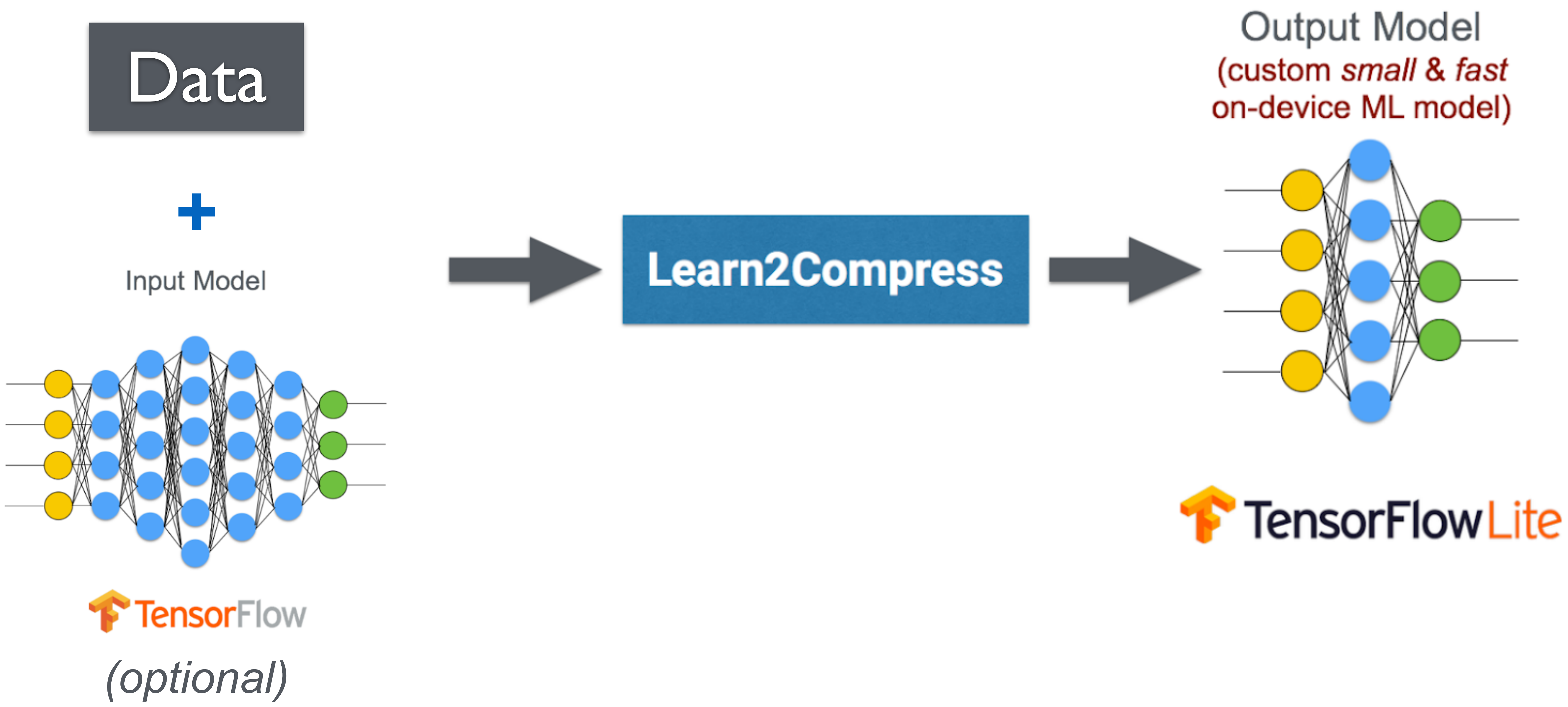
ProjectionNets for Language Tasks

Text classification results *(precision@1)*

Model	Compression *(wrt RNN)*	Smart Reply Intent	ATIS
Random (Kannan et al., 2016)	-	5.2	-
Frequency (Kannan et al., 2016)	-	9.2	72.2
LSTM (Kannan et al., 2016)	1	96.8	-
Attention RNN (Liu & Lane, 2016)	1	-	91.1
ProjectionNet **(our approach)** *[T=70,d=14]→FC [256 x 128]*	>10	**97.7**	**91.3**

- Efficient wrt compute/memory while maintaining high quality
  - ➡ On ATIS, ProjectionNet (quantized) achieves **91.0%** with tiny footprint (**285KB**)
- Achieves SoTA for NLP tasks

Learn2Compress: Build your own On-Device Models



Google AI Blog “Custom On-Device ML Models with Learn2Compress”

Thank You!

http://www.sravi.org

@ravisujith

Paper *Efficient On-Device Models using Neural Projections*
http://proceedings.mlr.press/v97/ravi19a.html

Check out our Workshop
Fri, Jun 14 (Room 203)

Joint Workshop on On-Device Machine Learning & Compact Deep Neural Network Representations (ODML-CDNNR)