

Conditional Networks

Unifying Deep Neural Networks and Decision Forests

Y. Ioannou, D. Robertson, D. Zikic, P. Kontschieder, J. Shotton, M. Brown, A. Criminisi



Outline

- **Introduction**
 - Trees, DAGs and deep neural networks (DNNs)
 - *Motivation:* Representing trees/DAGs as MLPs
 - *Motivation:* ReLUs and data routing
- **Conditional networks**
 - The model
 - Training via back-propagation
- **Experiments and results**
 - Conditional sparsification of a perceptron
 - Comparing architectures on ImageNet
 - Comparing architectures on CIFAR
 - Conditional ensembles
- **Conclusion**



Outline

- **Introduction**
 - Trees, DAGs and deep neural networks (DNNs)
 - *Motivation:* Representing trees/DAGs as MLPs
 - *Motivation:* ReLUs and data routing
- **Conditional networks**
 - The model
 - Training via back-propagation
- **Experiments and results**
 - Conditional sparsification of a perceptron
 - Comparing architectures on ImageNet
 - Comparing architectures on CIFAR
 - Conditional ensembles
- **Conclusion**



Deep learning and jungles in Microsoft products

Deep neural networks in speech recognition



... in Skype universal translator ...

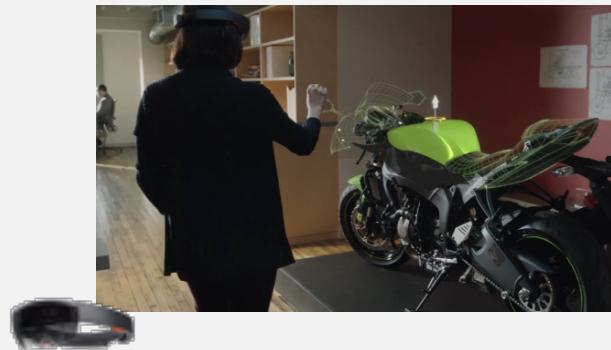


... see also work on ImageNet, project Adam ...

Decision forests in Kinect's body tracking



Decision jungles in HoloLens' gesture recognition



... also in Azure ML

Deep learning and jungles in Microsoft products

Deep neural networks in speech recognition



... in Skype universal translator ...



... see also work on ImageNet, project Adam ...

Decision forests in Kinect's body tracking



Decision jungles in secret projects

SECRET

... also in Azure ML

Deep learning and jungles in Microsoft products

Deep neural networks in speech recognition



... in Skype universal translator ...



... see also work on ImageNet, project Adam ...

Decision forests in Kinect's body tracking

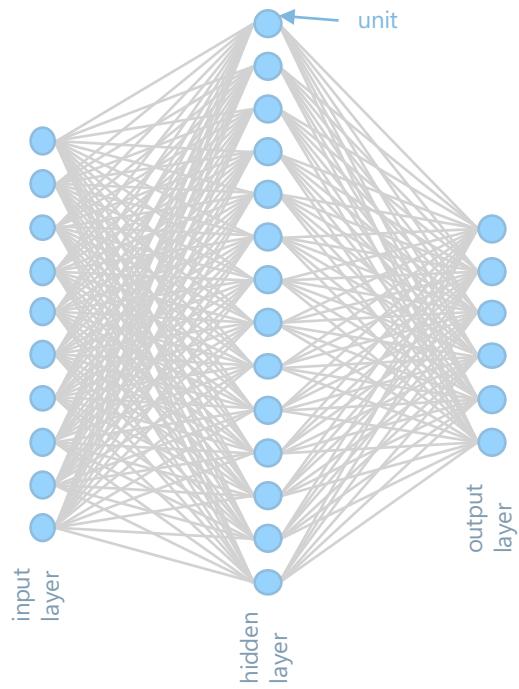


Decision jungles in Azure ML

... also in secret projects

Neural networks vs decision trees

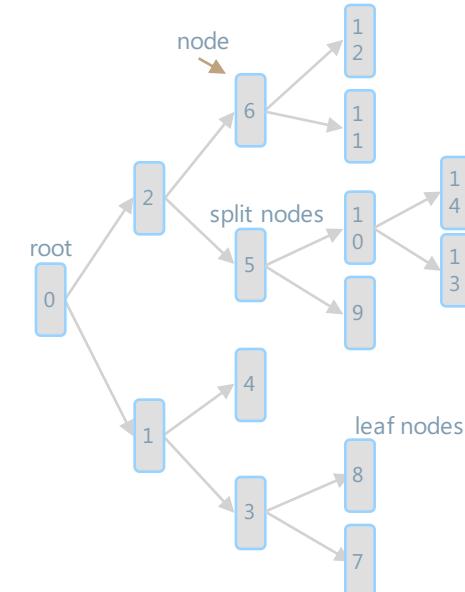
Multi-Layer Perceptron



Some practical differences

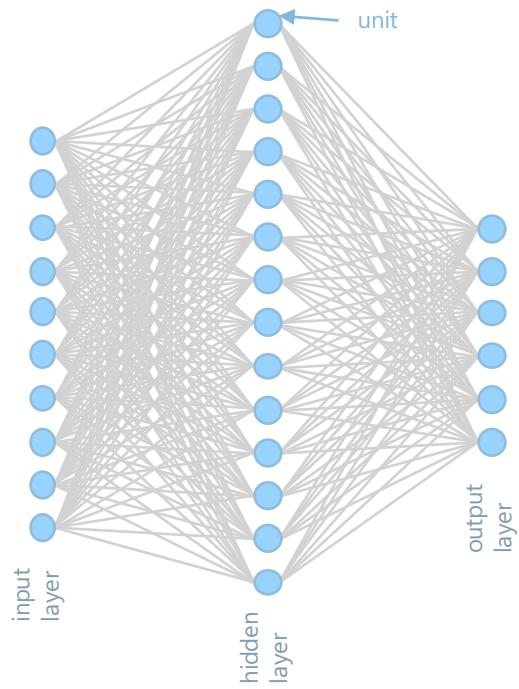
	ML Perceptrons	Decision Trees/DAGs
locality of computation	All "neurons" involved	Conditional computation
data transformation	Projections, non-linearities	Identity, copy
training algorithm	Gradient descent, back-prop	Greedy, randomized
structure	fixed ahead of time	learned and optimized
output	soft-max probabilities	histograms, probabilities
...

Decision Tree



Neural networks vs decision trees

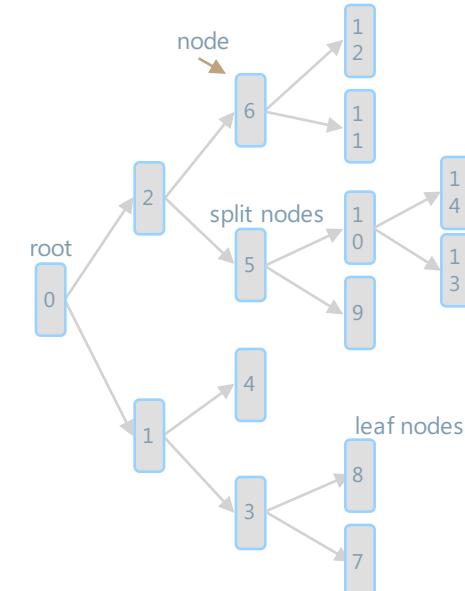
Multi-Layer Perceptron



Some practical differences

	ML Perceptrons	Decision Trees/DAGs
locality of computation	All "neurons" involved	Conditional computation
data transformation	Projections, non-linearities	Identity, copy
training algorithm	Gradient descent, back-prop	Greedy, randomized
structure	fixed ahead of time	learned and optimized
output	soft-max probabilities	histograms, probabilities
...

Decision Tree



Outline

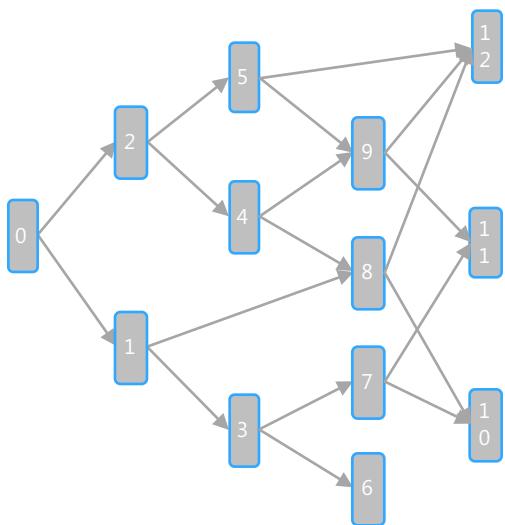
- **Introduction**
 - Trees, DAGs and deep neural networks (DNNs)
 - *Motivation: Representing trees/DAGs as MLPs*
 - *Motivation: ReLUs and data routing*
- **Conditional networks**
 - The model
 - Training via back-propagation
- **Experiments and results**
 - Conditional sparsification of a perceptron
 - Comparing architectures on ImageNet
 - Comparing architectures on CIFAR
 - Conditional ensembles
- **Conclusion**



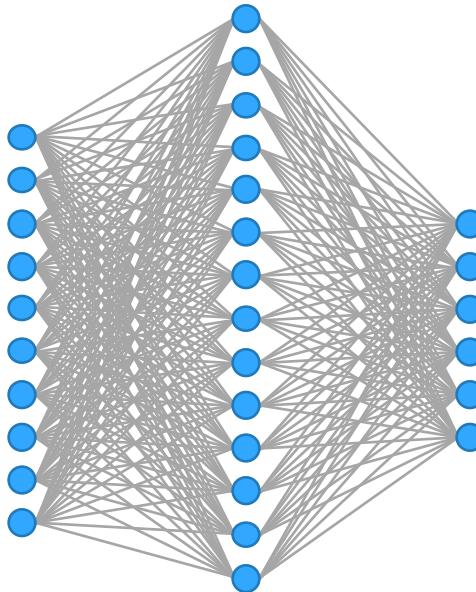
Representing trees as perceptrons

Proposition. Any DAG can be represented as a two-layer perceptron.

Directed Acyclical Graph (DAG)



Two-Layer Perceptron

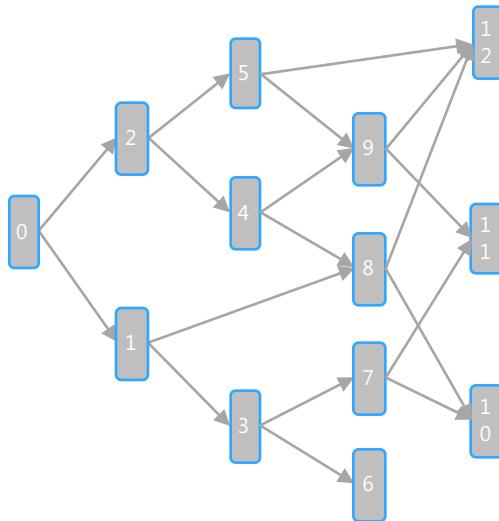


Representing trees as perceptrons

Proposition. Any DAG can be represented as a two-layer perceptron.

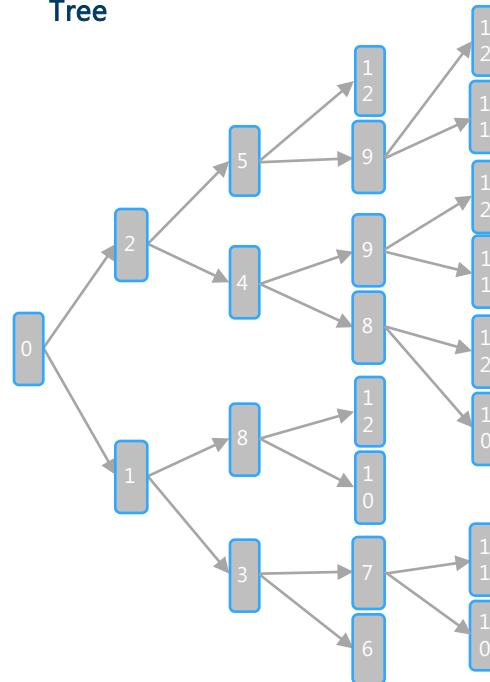
Proof step 1 – any DAG can be represented as a tree

Directed Acyclical Graph



1

Tree



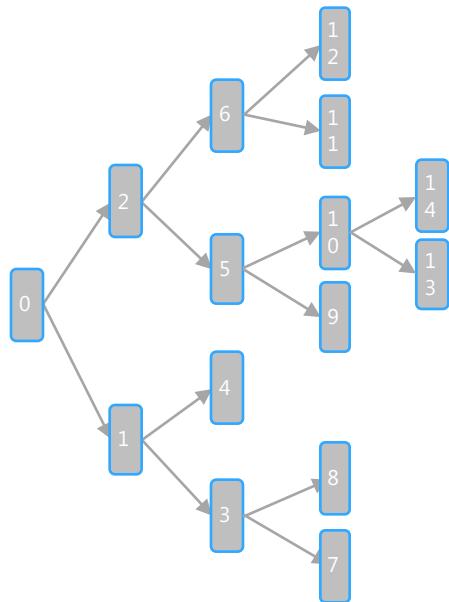
Similarly, n-ary trees can be transformed into binary trees.

Representing trees as perceptrons

Proposition. Any DAG can be represented as a two-layer perceptron.

Proof step 2 – trees can be represented as 2-layer MLPs

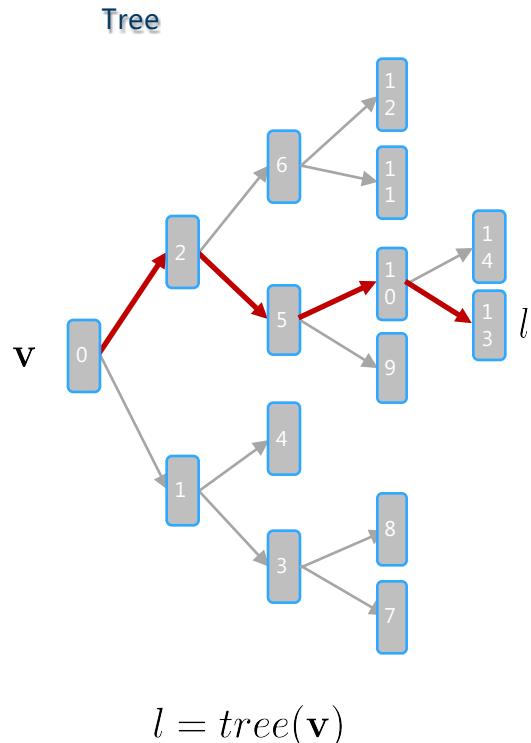
Tree



Representing trees as perceptrons

Proposition. Any DAG can be represented as a two-layer perceptron.

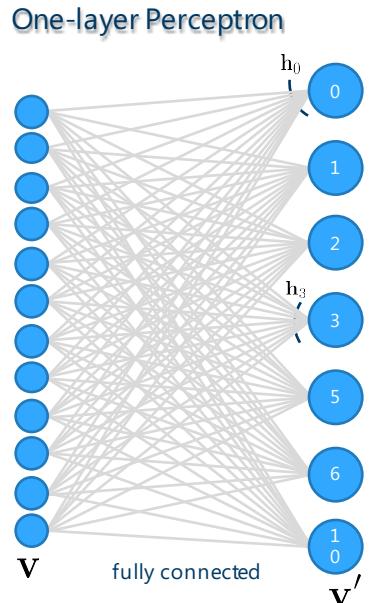
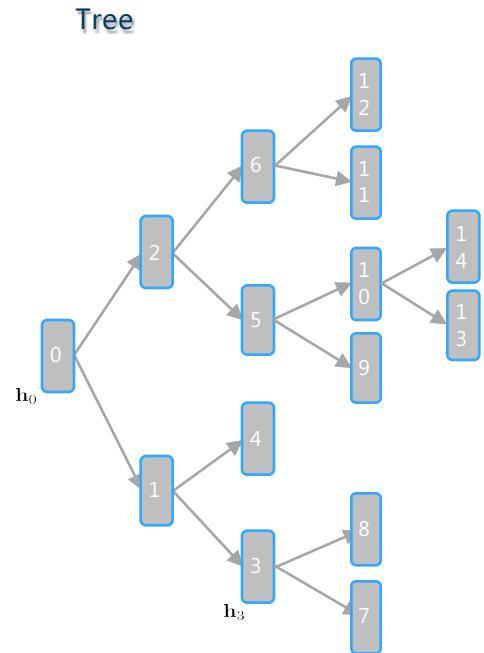
Proof step 2 – trees can be represented as 2-layer MLPs



Representing trees as perceptrons

Proposition. Any DAG can be represented as a two-layer perceptron.

Proof step 2 – trees can be represented as 2-layer MLPs

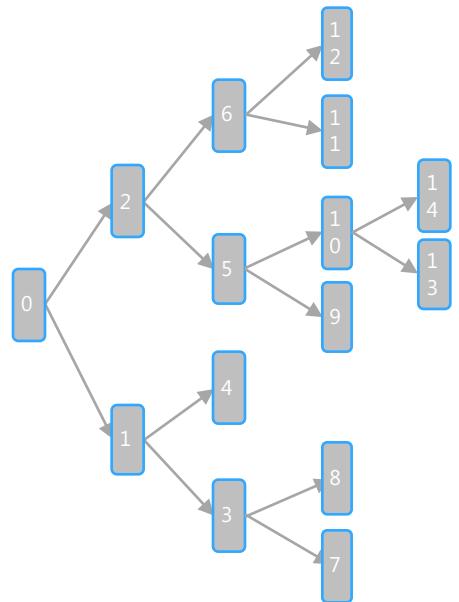


Representing trees as perceptrons

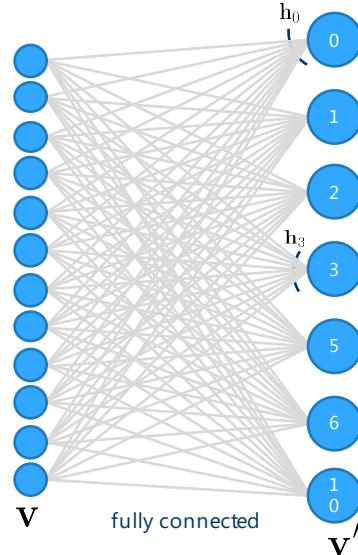
Proposition. Any DAG can be represented as a two-layer perceptron.

Proof step 2 – trees can be represented as 2-layer MLPs

Tree



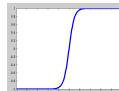
One-layer Perceptron



Hidden activations

$$v'_j = 2(\sigma(v \cdot h_j; 0, \beta) - 0.5) \in [-1, 1]$$

$$\sigma(x; \mu, \beta) = \frac{1}{1 + e^{\frac{\mu-x}{\beta}}}$$

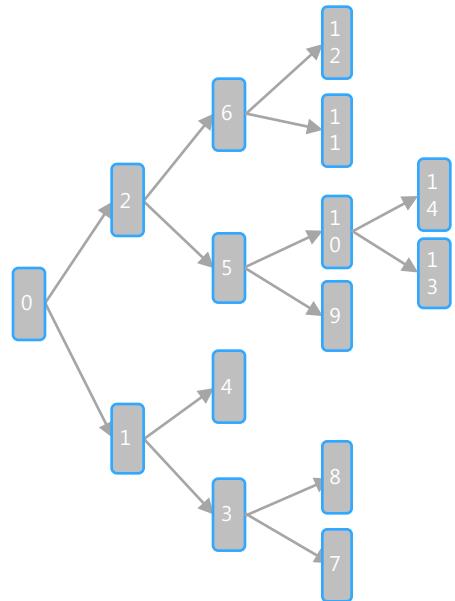


Representing trees as perceptrons

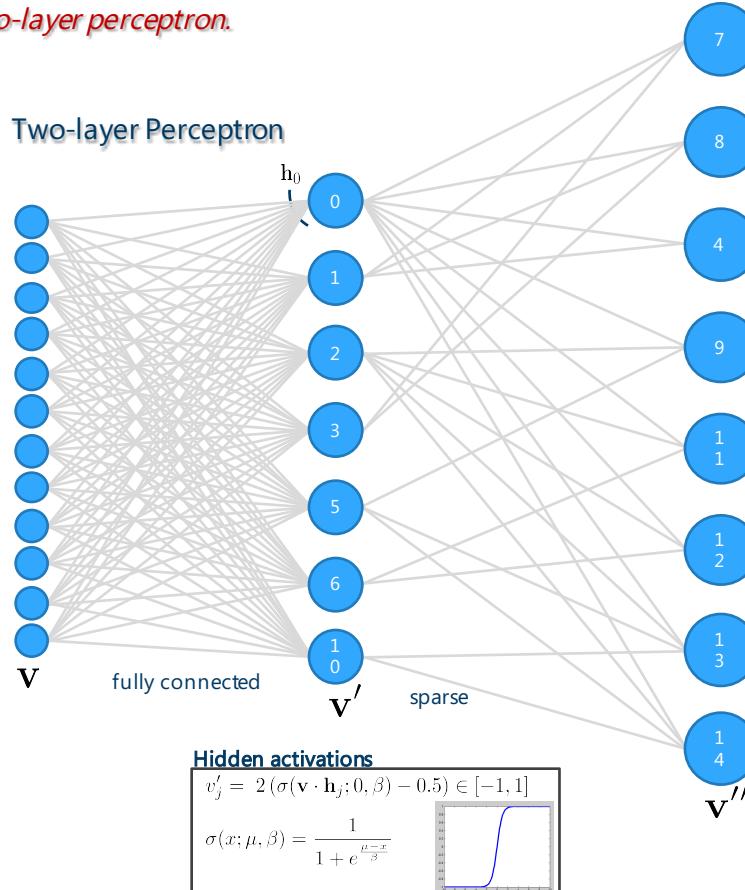
Proposition. Any DAG can be represented as a two-layer perceptron.

Proof step 2 – trees can be represented as 2-layer MLPs

Tree



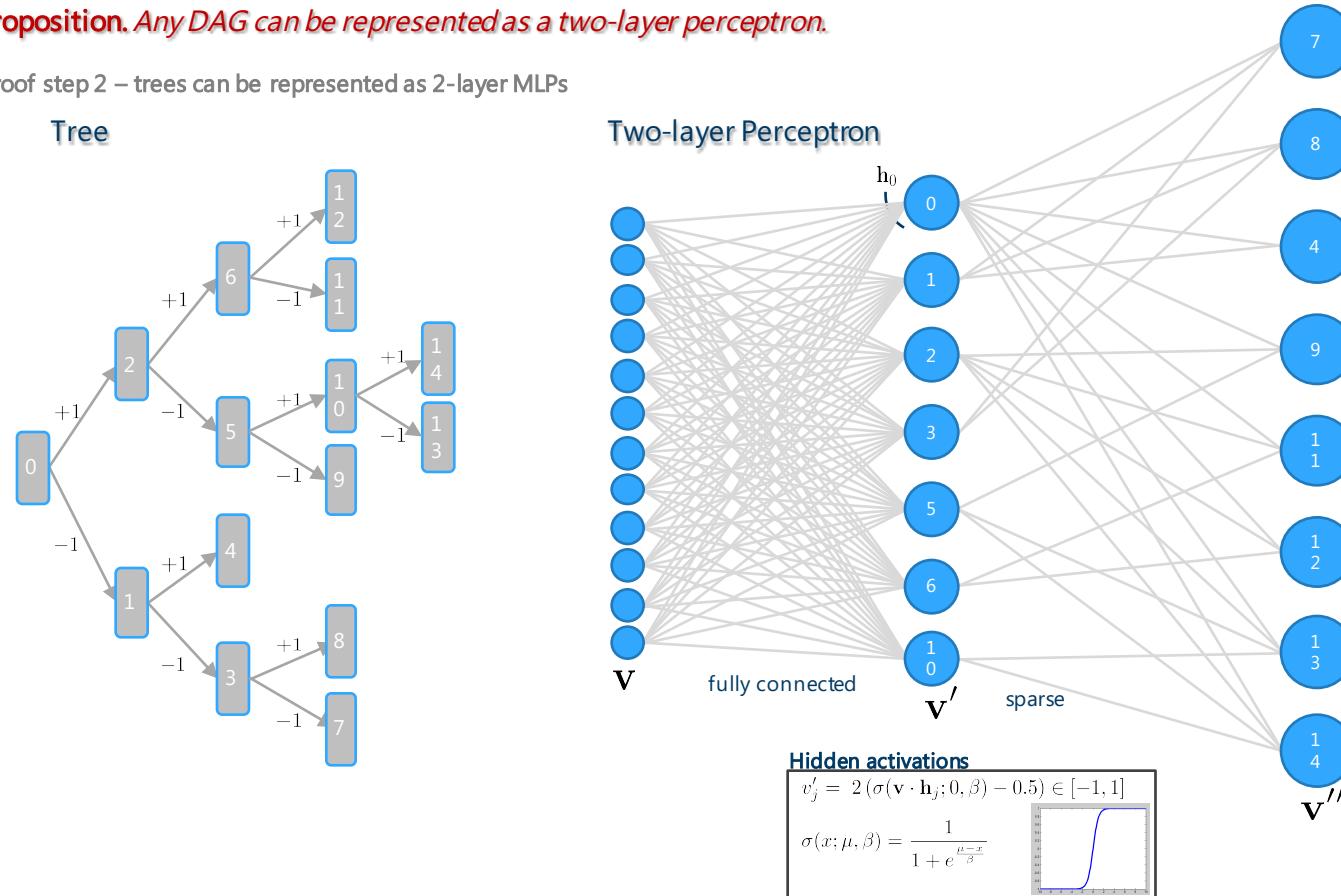
Two-layer Perceptron



Representing trees as perceptrons

Proposition. Any DAG can be represented as a two-layer perceptron.

Proof step 2 – trees can be represented as 2-layer MLPs

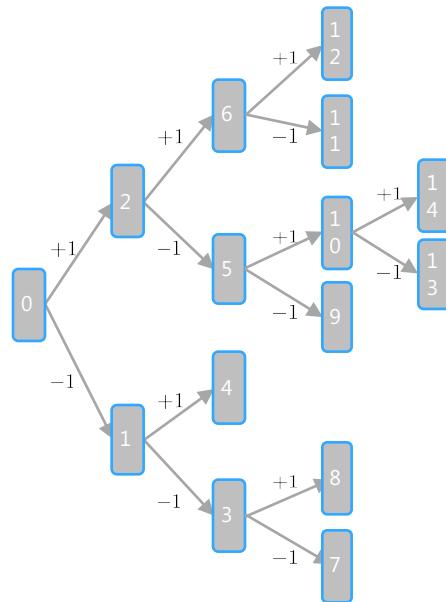


Representing trees as perceptrons

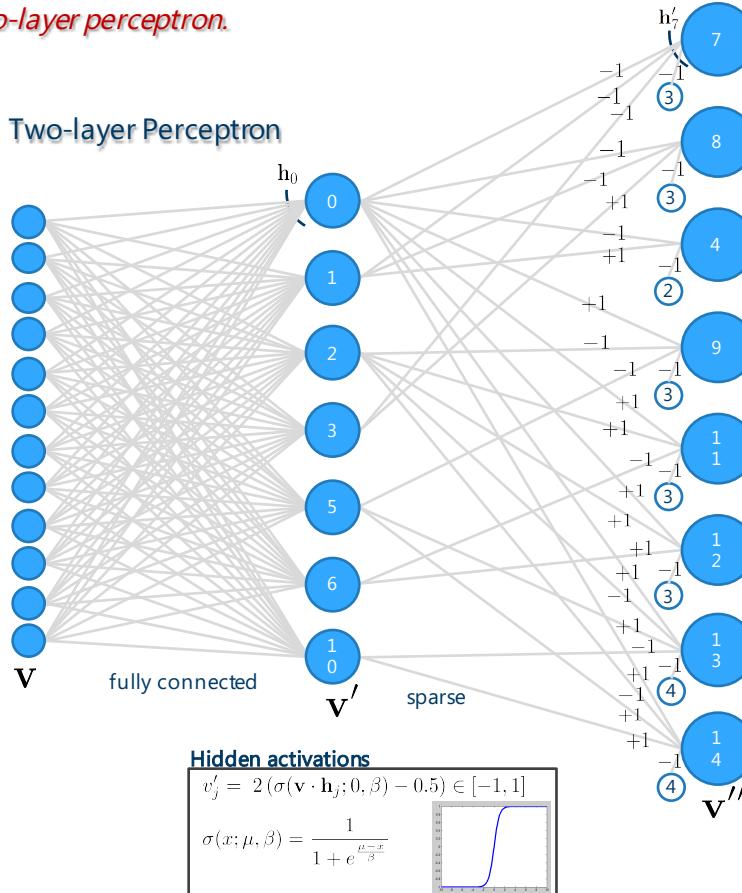
Proposition. Any DAG can be represented as a two-layer perceptron.

Proof step 2 – trees can be represented as 2-layer MLPs

Tree



Two-layer Perceptron

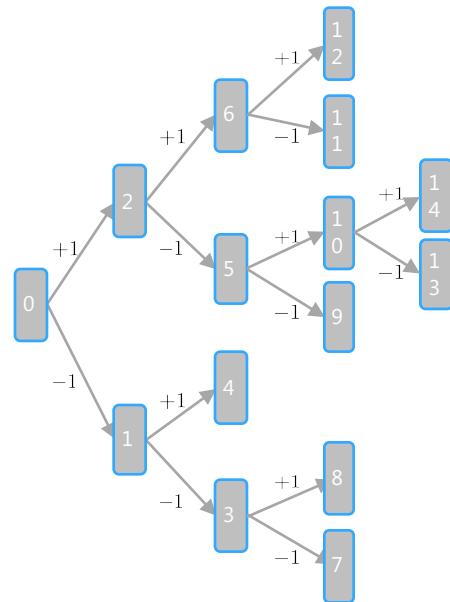


Representing trees as perceptrons

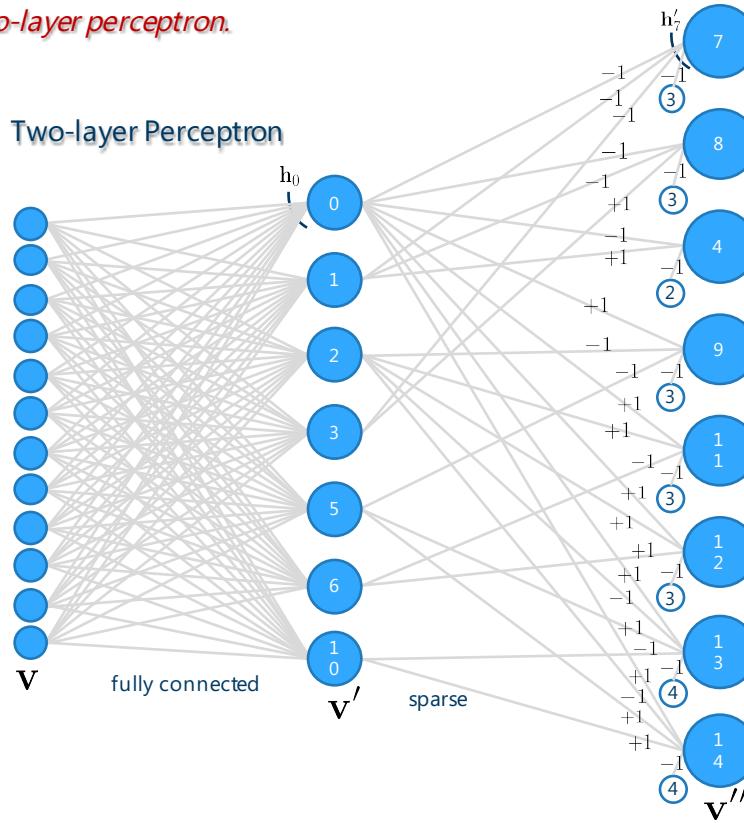
Proposition. Any DAG can be represented as a two-layer perceptron.

Proof step 2 – trees can be represented as 2-layer MLPs

Tree



Two-layer Perceptron

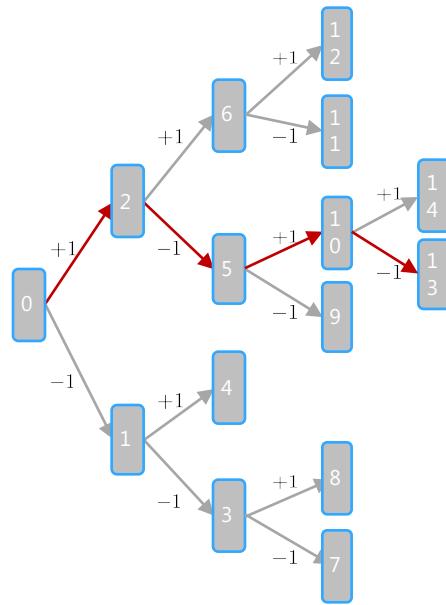


Representing trees as perceptrons

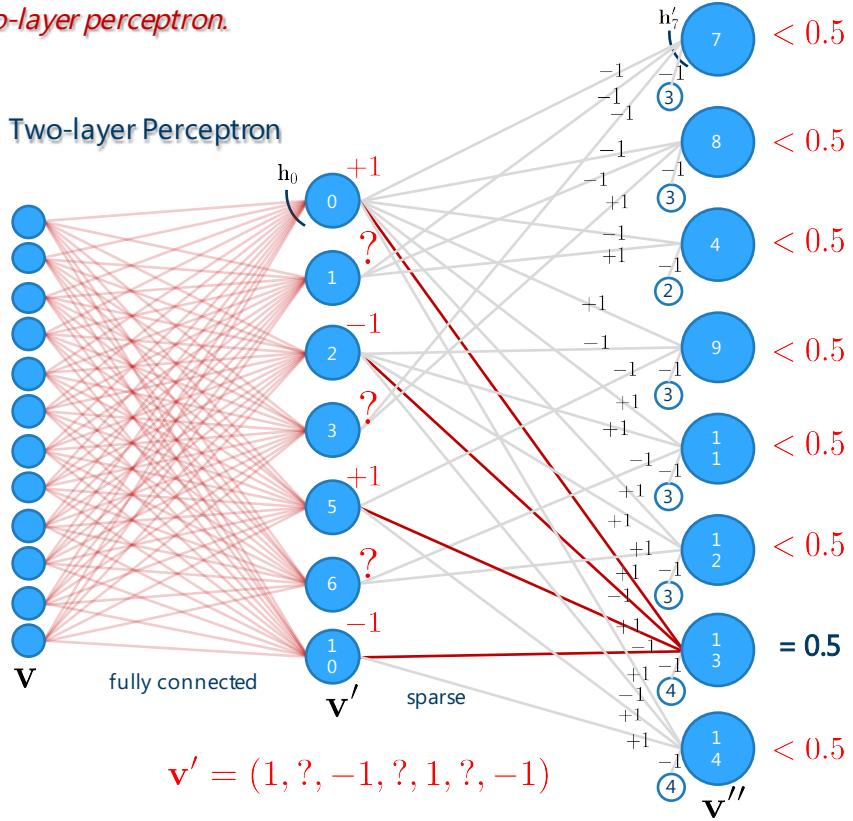
Proposition. Any DAG can be represented as a two-layer perceptron.

Proof step 2 – trees can be represented as 2-layer MLPs

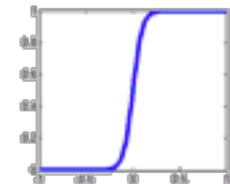
Tree



Two-layer Perceptron



$$\leq \frac{1}{1 + \exp(-\frac{1}{b})}$$



$$v''_j = \sigma(v' \cdot h'_j; 0, \beta') \in [0, 1]$$

$$\sigma(x; \mu, \beta) = \frac{1}{1 + e^{\frac{\mu-x}{\beta}}}$$

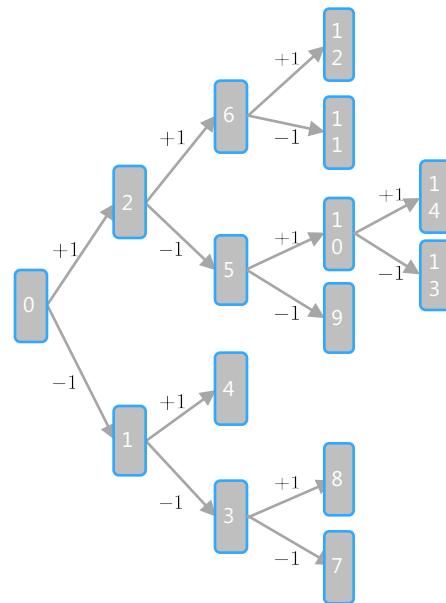
Representing trees as perceptrons

(script_equivalence_test_4.m)

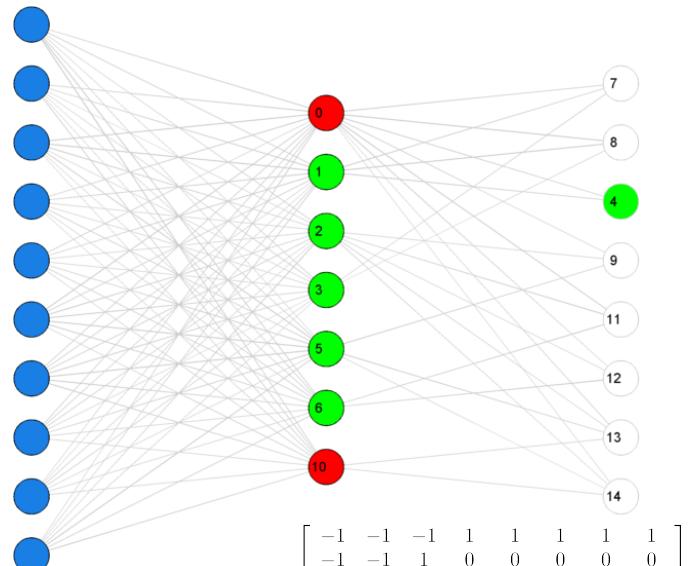
Proposition. Any DAG can be represented as a two-layer perceptron.

Proof step 2 – trees can be represented as 2-layer MLPs

Tree



Conditional Perceptron (i.e. a special 2-layer MLP)



$$v'_j = 2(\sigma(\mathbf{v} \cdot \mathbf{h}_j; 0, \beta) - 0.5) \in [-1, 1]$$

$$\beta = 0.01$$

$$v''_j = \sigma(\mathbf{v}' \cdot \mathbf{h}'_j; 0, \beta') \in [0, 1]$$

$$\beta' = 0.01$$

A tree/Dag can be thought of as a sparsified version of a 2-layer perceptron

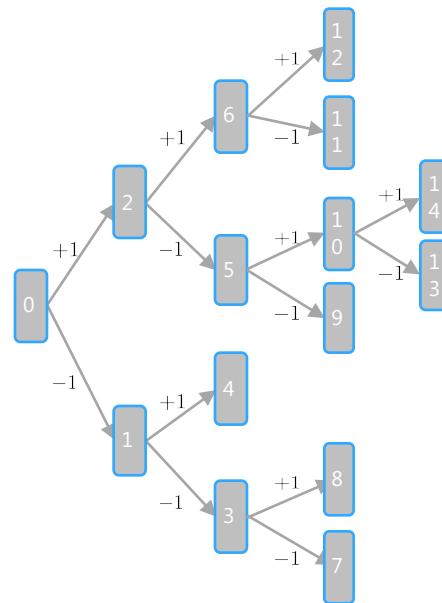
Representing trees as perceptrons

(script_equivalence_test_4.m)

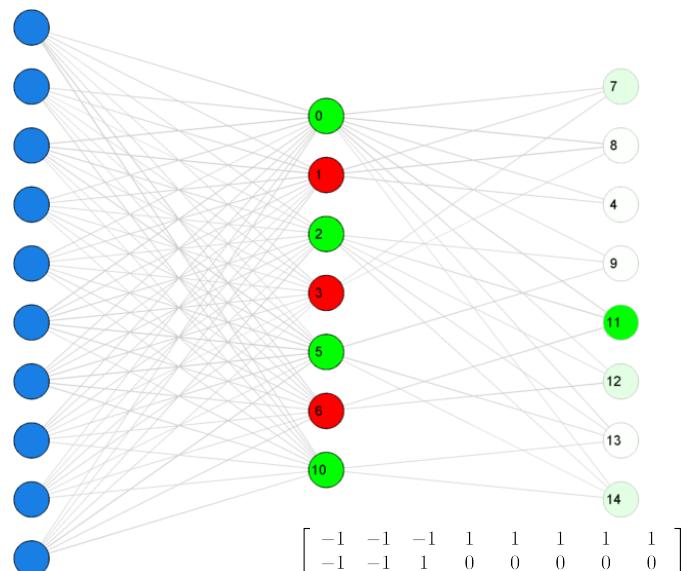
Proposition. Any DAG can be represented as a two-layer perceptron.

Proof step 2 – trees can be represented as 2-layer MLPs

Tree



Conditional Perceptron (i.e. a special 2-layer MLP)



$$v'_j = 2(\sigma(\mathbf{v} \cdot \mathbf{h}_j; 0, \beta) - 0.5) \in [-1, 1]$$

$$\beta = 0.01$$

$$v''_j = \sigma(\mathbf{v}' \cdot \mathbf{h}'_j; 0, \beta') \in [0, 1]$$

$$\beta' = 0.7$$

A tree/Dag can be thought of as a sparsified version of a 2-layer perceptron

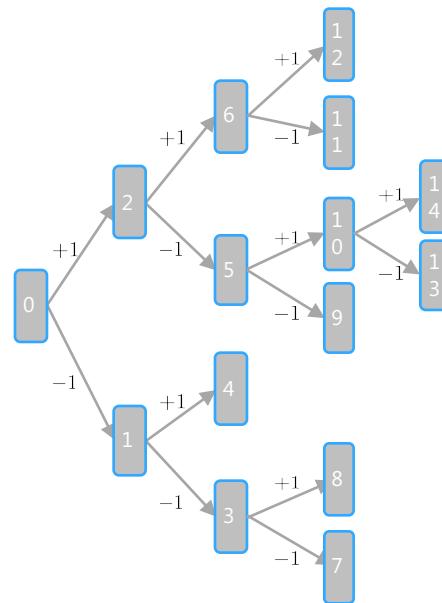
Representing trees as perceptrons

(script_equivalence_test_4.m)

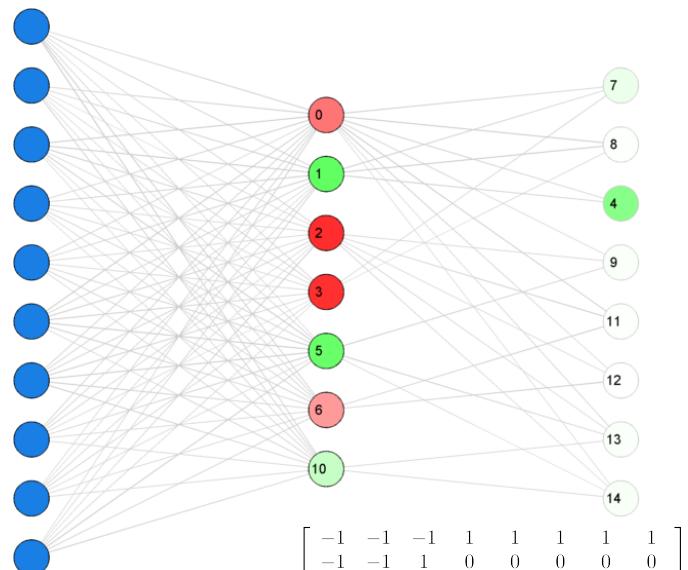
Proposition. Any DAG can be represented as a two-layer perceptron.

Proof step 2 – trees can be represented as 2-layer MLPs

Tree



Conditional Perceptron (i.e. a special 2-layer MLP)



$$v'_j = 2(\sigma(\mathbf{v} \cdot \mathbf{h}_j; 0, \beta) - 0.5) \in [-1, 1]$$

$$\beta = 0.7$$

$$v''_j = \sigma(\mathbf{v}' \cdot \mathbf{h}'_j; 0, \beta') \in [0, 1]$$

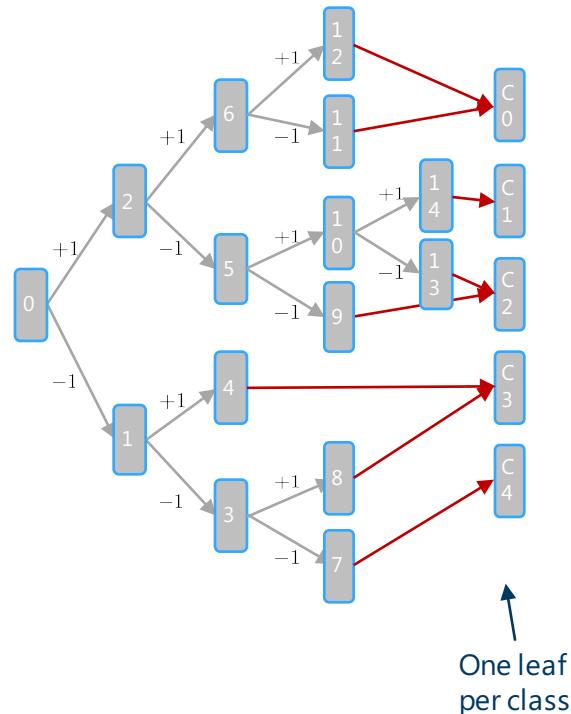
$$\beta' = 0.7$$

A tree/Dag can be thought of as a sparsified version of a 2-layer perceptron

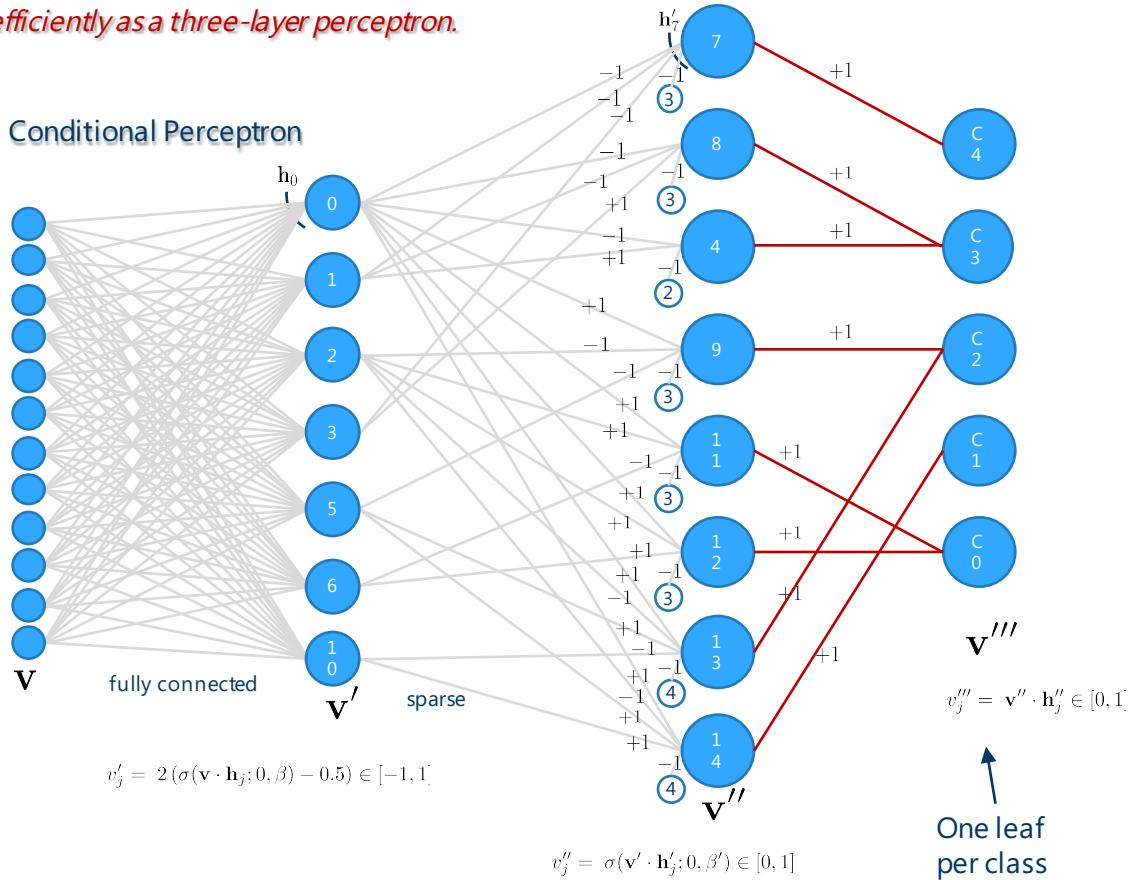
Representing trees as perceptrons

Corollary. Some useful DAGs can be represented efficiently as a three-layer perceptron.

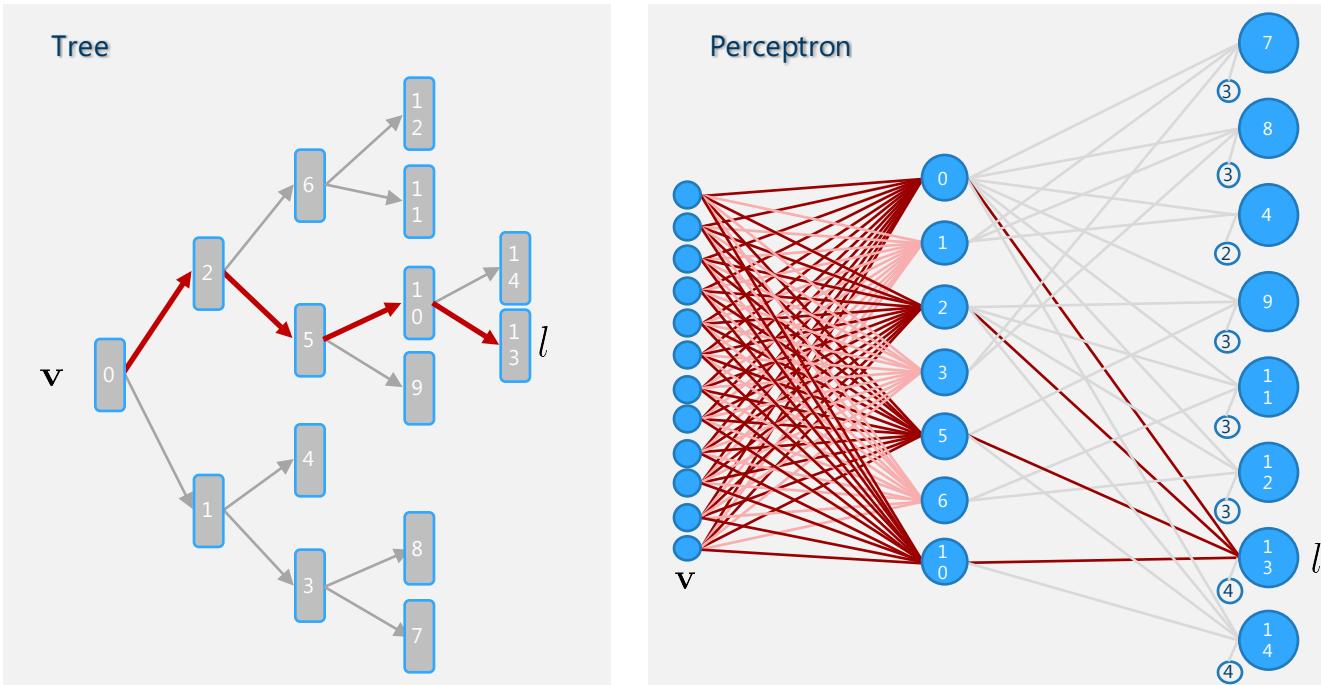
A special DAG



Conditional Perceptron



Representing trees as perceptrons



Functional equivalence

$$l = \text{tree}(\mathbf{v})$$

No computational equivalence

Conditional computation

$$l = \text{perceptron}(\mathbf{v})$$

No conditional computation

Goal of conditional networks: adding conditional computation to neural networks

Representing trees as perceptrons

Proposition. Any DAG can be represented as a two-layer perceptron.

Not a new idea. Rediscovering and revisiting the work in

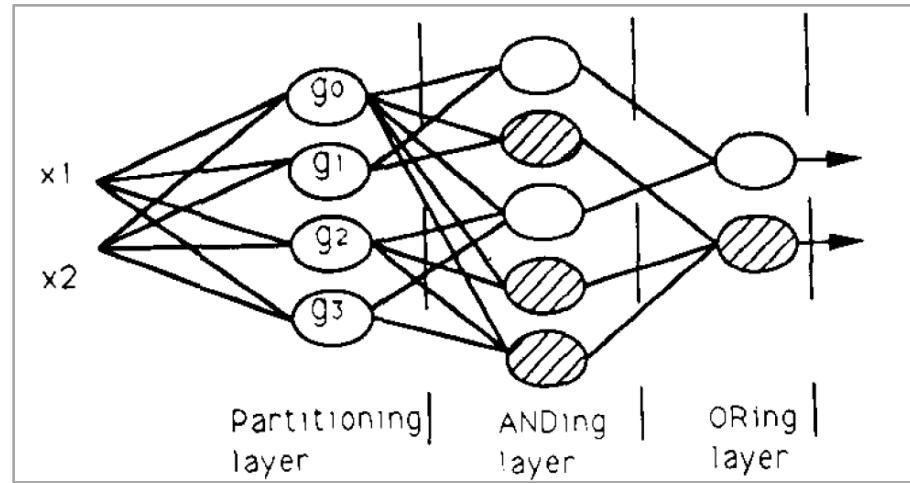
[Entropy Nets: From Decision Trees to Neural Networks.
I. K. Sethi. Proc. of the IEEE (Vol. 78 , Issue 10) . Oct 1990]

[Casting Random Forests as Artificial Neural Networks
(and Profiting from It). J. Welbl. Proc. of GCPR. 2014]

"restructuring a decision tree as a multilayer neural network"
"...automatic tree generation...incremental learning..."
"neural network design through decision tree mappings..."
"soft decision making"

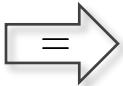
Training an MLP via entropy-based loss. What about training a tree with back-prop instead?
"...far fewer connections..." -> greater efficiency

"initializing an MLP from a random forest works better than initializing at random..."



Representing trees as perceptrons

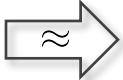
Tree / DAG



Sparse
Neural
Network



Deep
Neural
Network



approximate

Tree / DAG



What do we gain?
What do we lose?

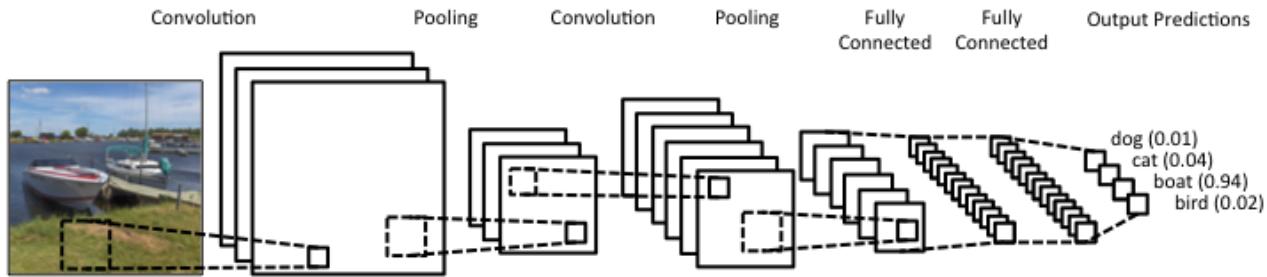
Outline

- **Introduction**
 - Trees, DAGs and deep neural networks (DNNs)
 - *Motivation:* Representing trees/DAGs as MLPs
 - *Motivation:* ReLUs and data routing
- **Conditional networks**
 - The model
 - Training via back-propagation
- **Experiments and results**
 - Conditional sparsification of a perceptron
 - Comparing architectures on ImageNet
 - Comparing architectures on CIFAR
 - Conditional ensembles
- **Conclusion**

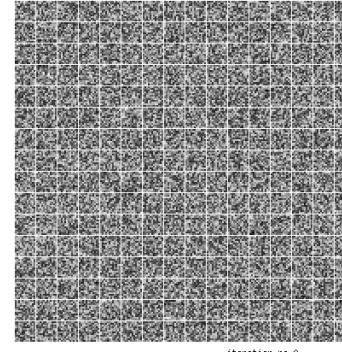


Structured activation sparsity and data routing

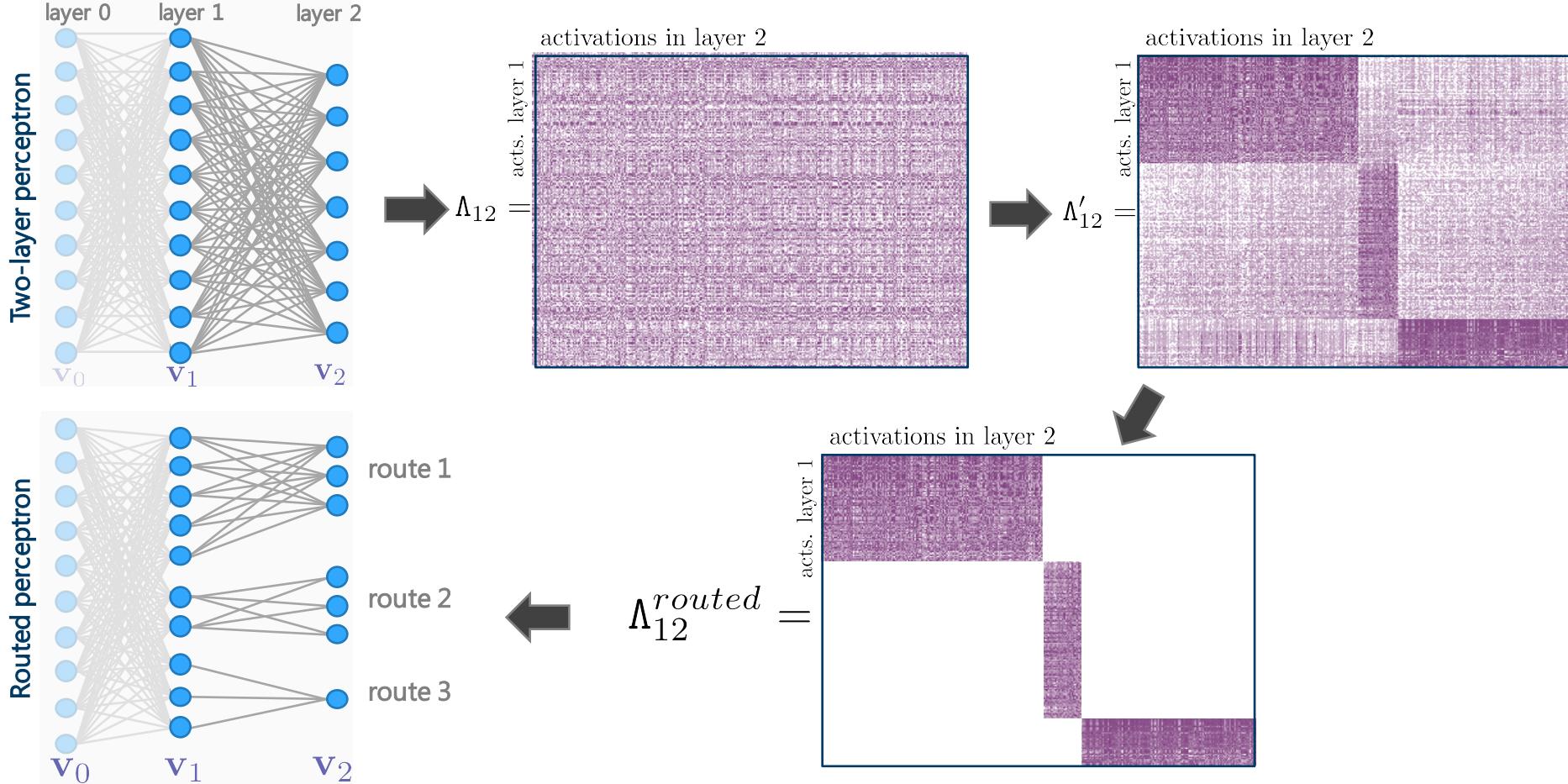
A deep convolutional neural network



Representation learning: selecting optimal convolution kernels



Structured activation sparsity and data routing



"Cells that fire together, wire together". [D. Hebb. *The organization of Behavior*. Wiley and Sons. 1949]

Trees and neural networks

What we have learnt so far

- A decision tree/DAG is a **special subclass** of multi-layer perceptrons.
- In general, at test time decision trees/DAGs perform **fewer computations** than MLPs because of data routing and conditional computation.
- However, hard data routing may yield **lower accuracy**. finally,
- Trees are **hard to train** in a globally optimal manner.

Therefore we introduce **conditional networks**

A *conditional network* is a new model that generalizes both (deep) neural networks and decision forests/jungles. Conditional networks marry efficient test-time **conditional computation** with the **accuracy** and **global trainability** of neural networks.

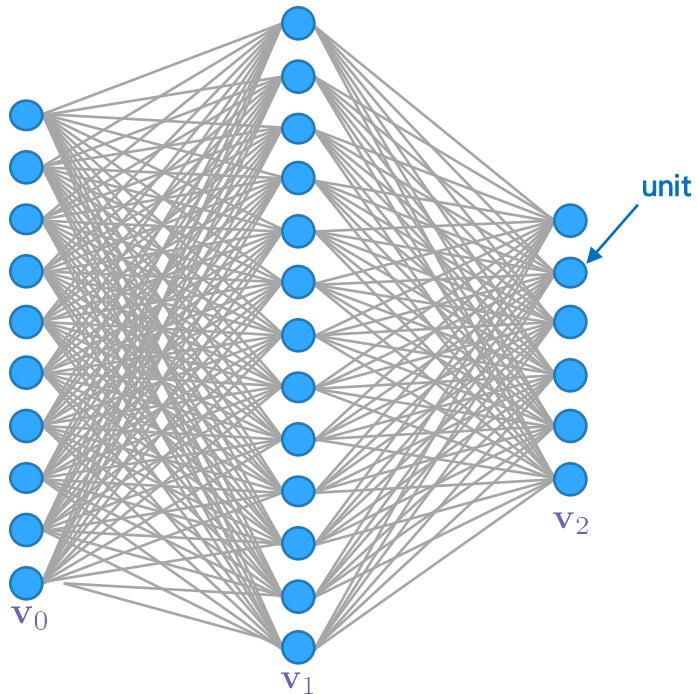
Outline

- **Introduction**
 - Trees, DAGs and deep neural networks (DNNs)
 - *Motivation:* Representing trees/DAGs as MLPs
 - *Motivation:* ReLUs and data routing
- **Conditional networks**
 - The model
 - Training via back-propagation
- **Experiments and results**
 - Conditional sparsification of a perceptron
 - Comparing architectures on ImageNet
 - Comparing architectures on CIFAR
 - Conditional ensembles
- **Conclusion**

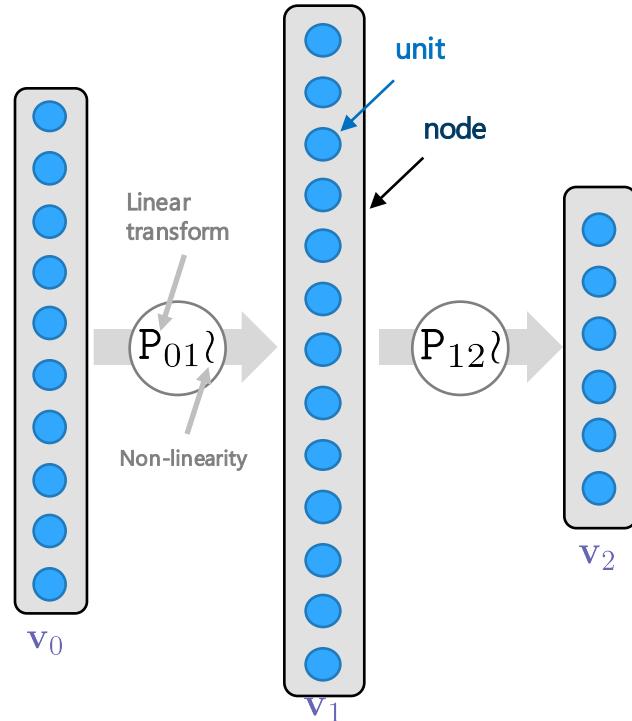


A new, compact graphical notation - MLP

Old notation for a 2-layer perceptron



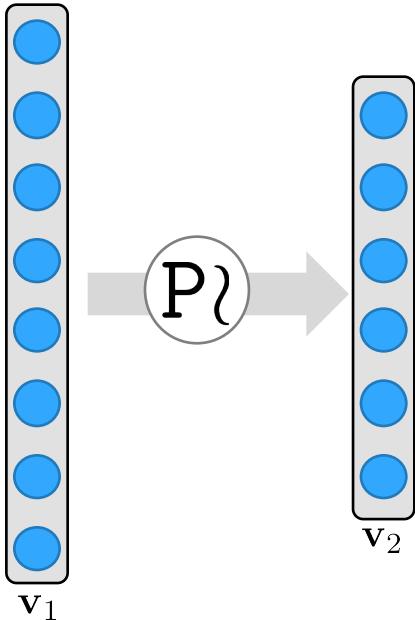
New notation



The non-linearity could be a ReLU, sigmoid, tanh...

A new, compact graphical notation - MLP

New notation



Algebraically

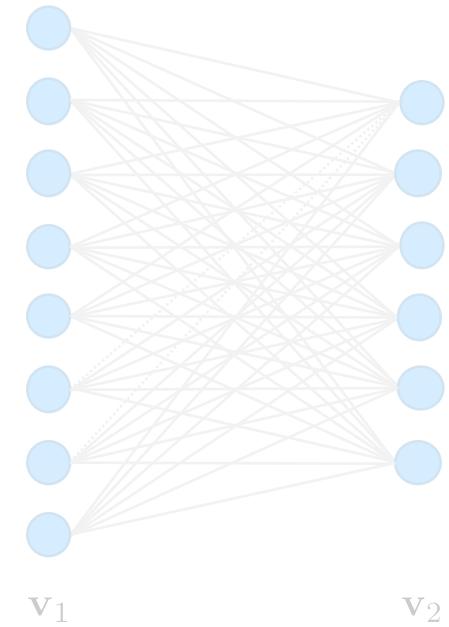
$$v_2 = \sigma(Pv_1)$$

Generic non-linearity

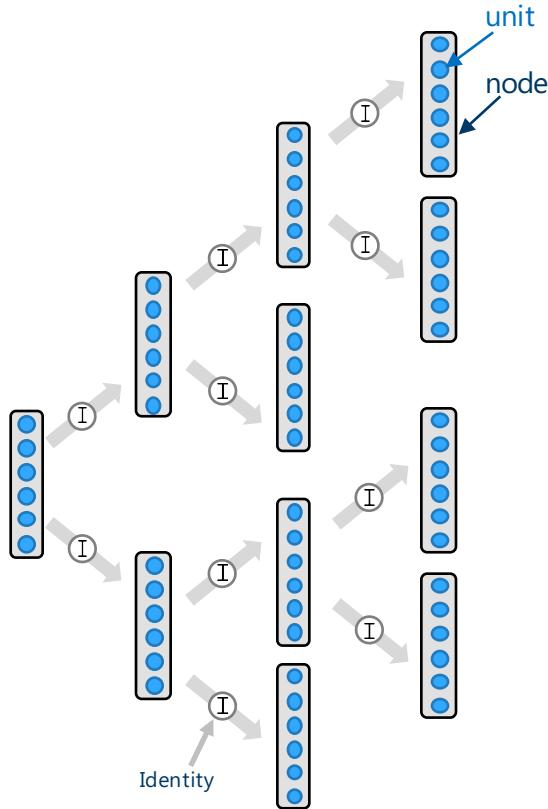
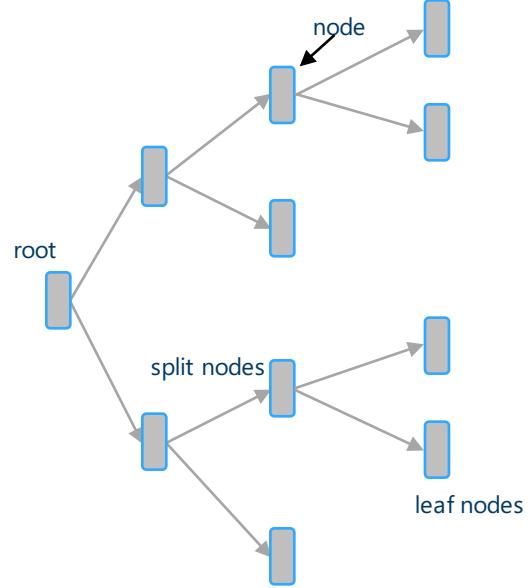
$$P = \begin{bmatrix} - & h_1 & - \\ - & h_2 & - \\ - & h_3 & - \\ - & h_4 & - \\ - & h_5 & - \\ - & h_6 & - \end{bmatrix}$$

The bias is incorporated within the Homogeneous notation for simplicity

Old notation



A new, compact graphical notation - Tree

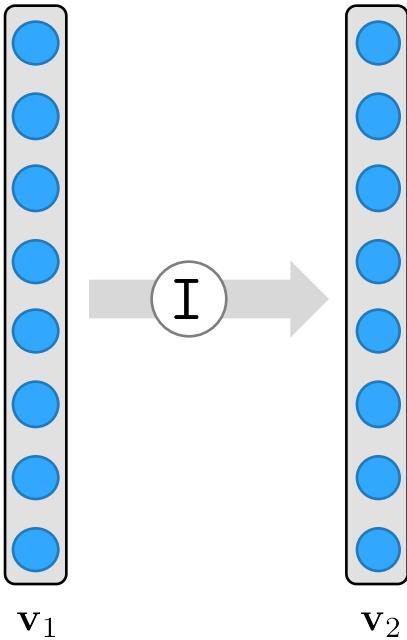


In a tree/DAG data is **moved** about, unchanged

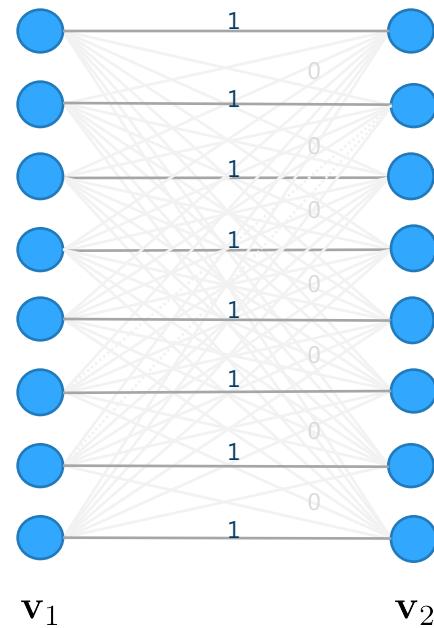
We need one more ingredient to make it all work...

A new, compact graphical notation

New notation



Old notation

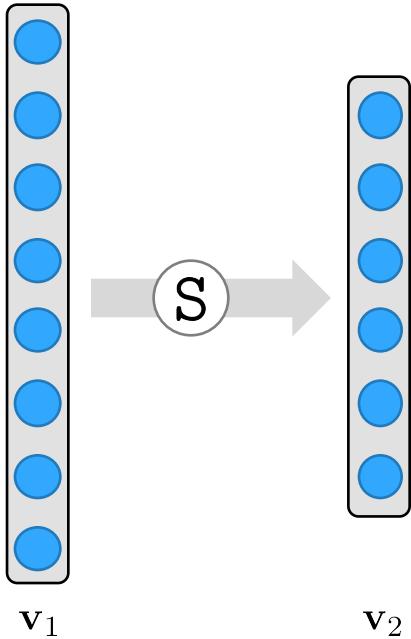


Algebraically

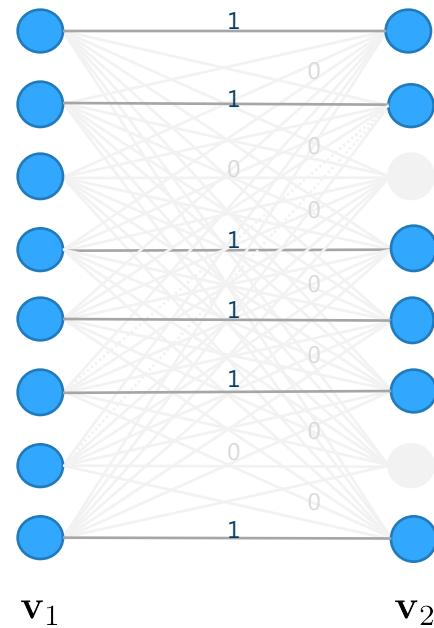
$$\mathbf{v}_2 = I\mathbf{v}_1$$

A new, compact graphical notation

New notation



Old notation

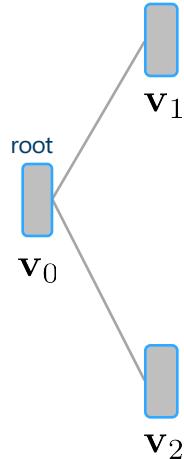


Algebraically

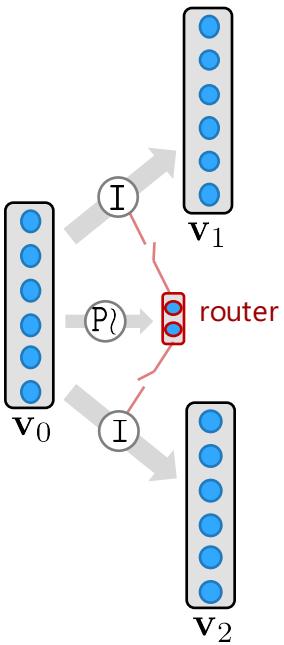
$$\mathbf{v}_2 = S\mathbf{v}_1$$

A new, compact graphical notation - tree

Old notation



New notation



Algebraically

Data transformation

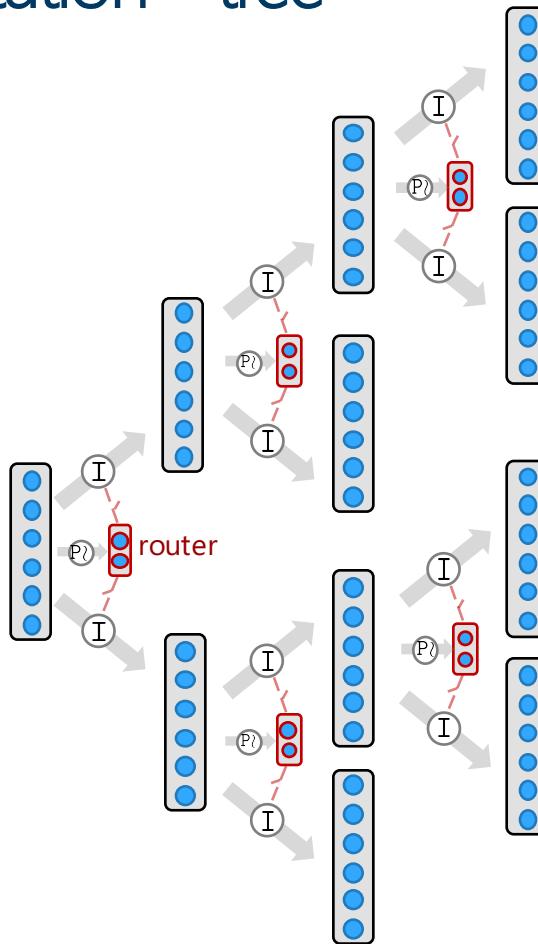
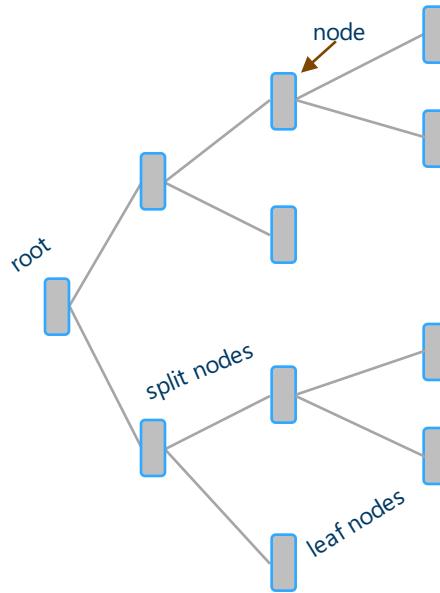
$$\mathbf{v}_i = r_i \mathbf{I} \mathbf{v}_0 \quad i \in \{1, 2\}$$

Routing weights (with ReLU non-linearity)

$$\mathbf{r} = \sigma(\mathbf{P} \mathbf{v}_0) \quad r_i \in [0, 1]$$

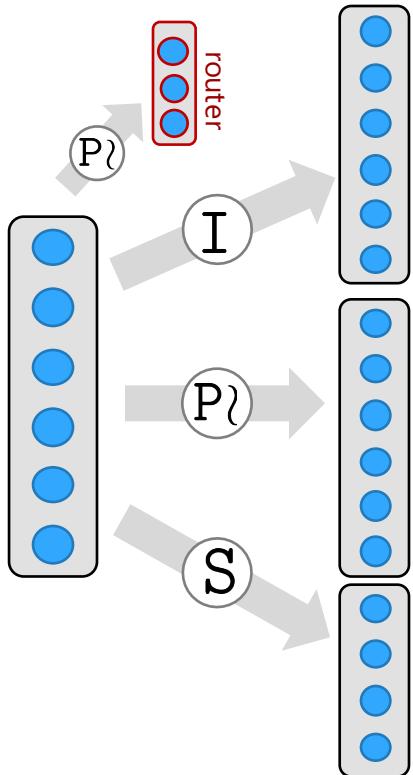
The router can produce $N >= 2$ outputs.

A new, compact graphical notation - tree

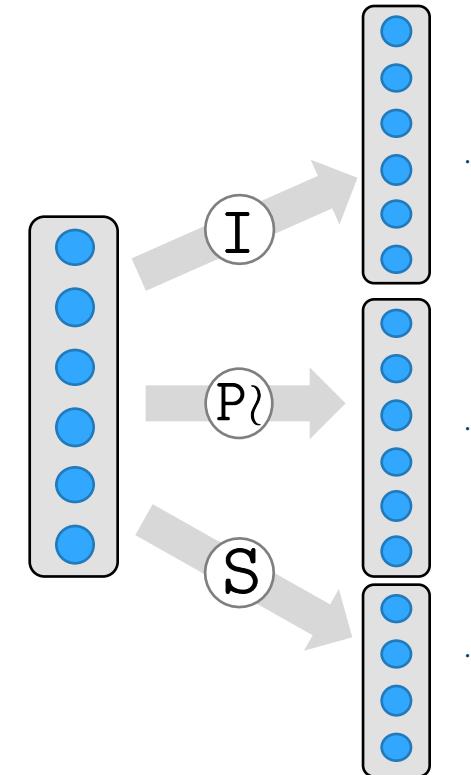


Explicit vs Implicit Data Routing

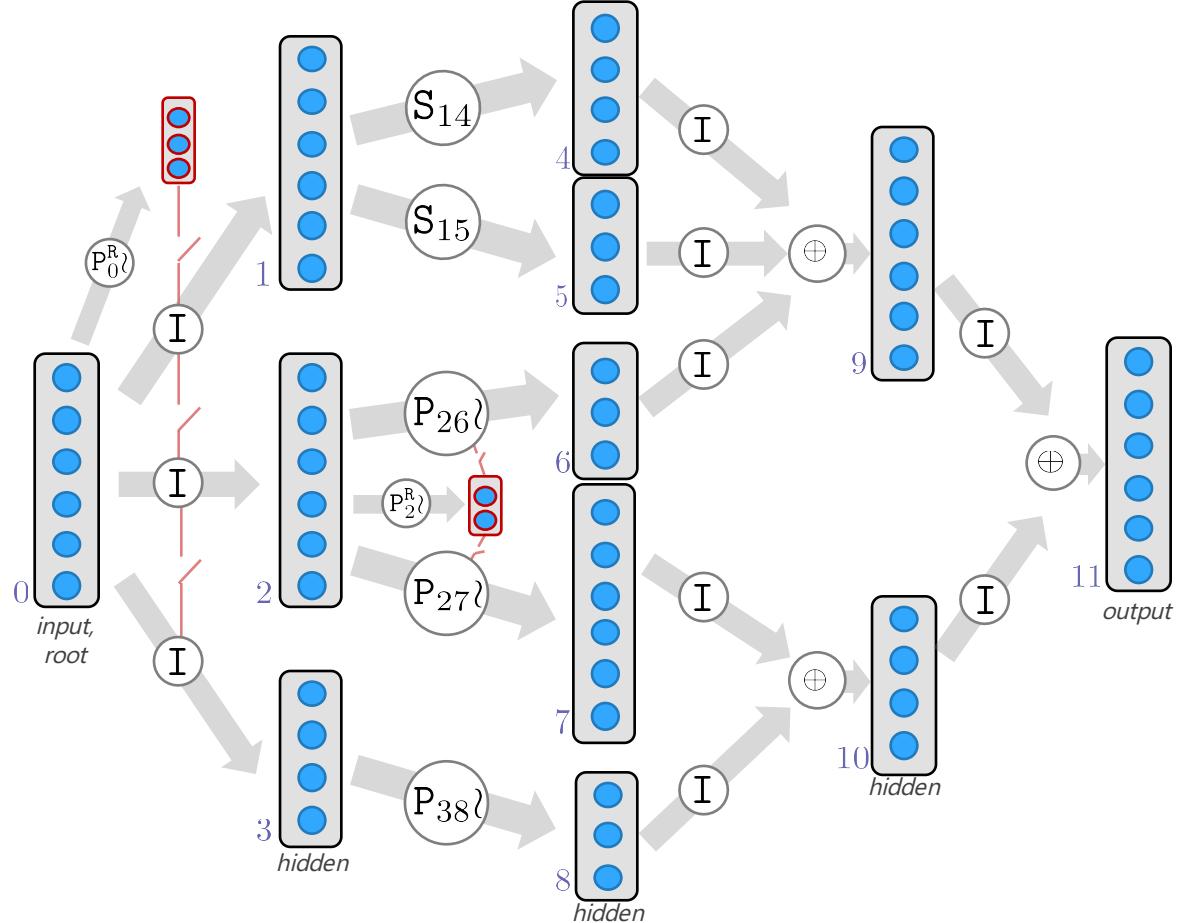
Explicit routing



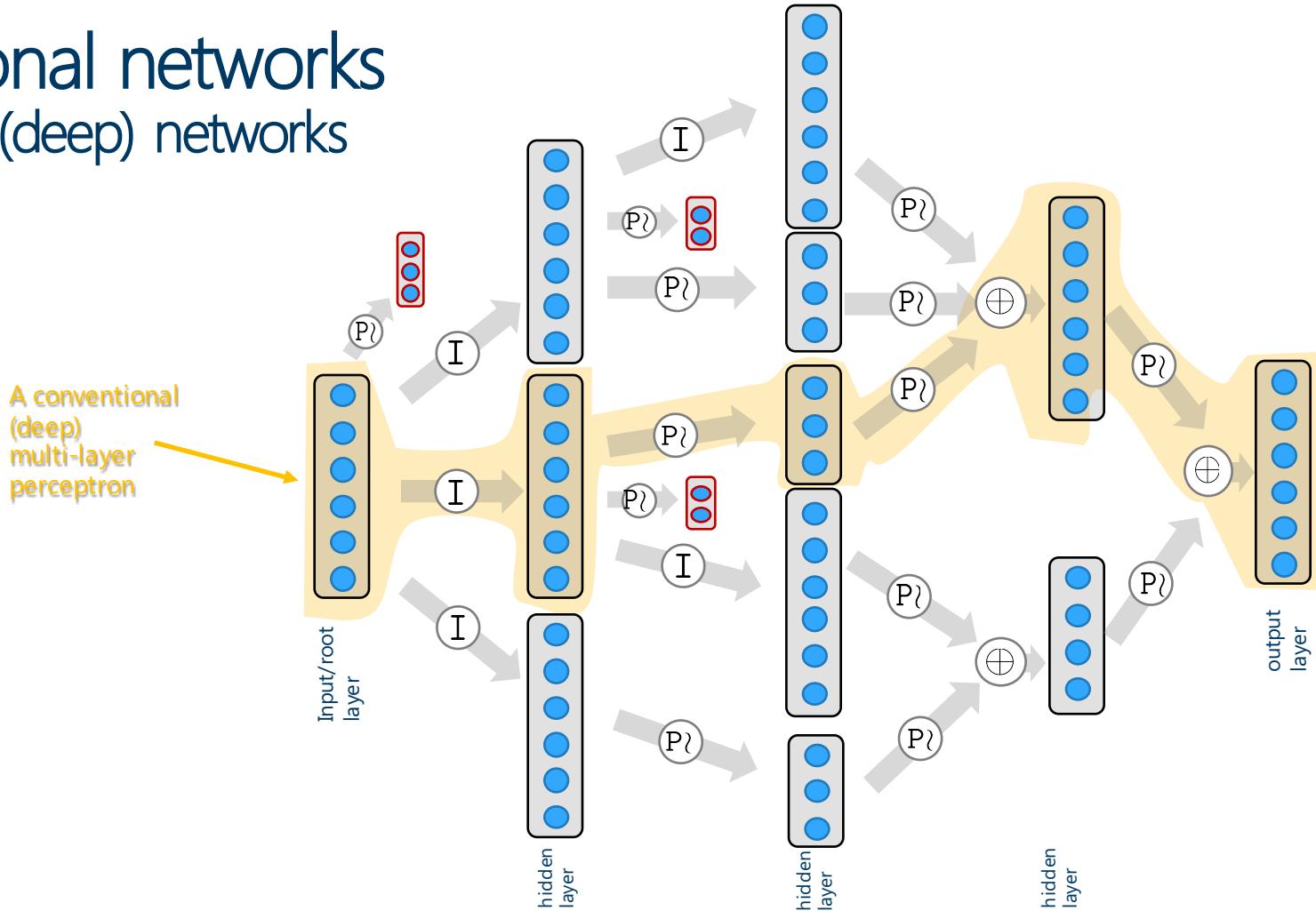
Implicit routing



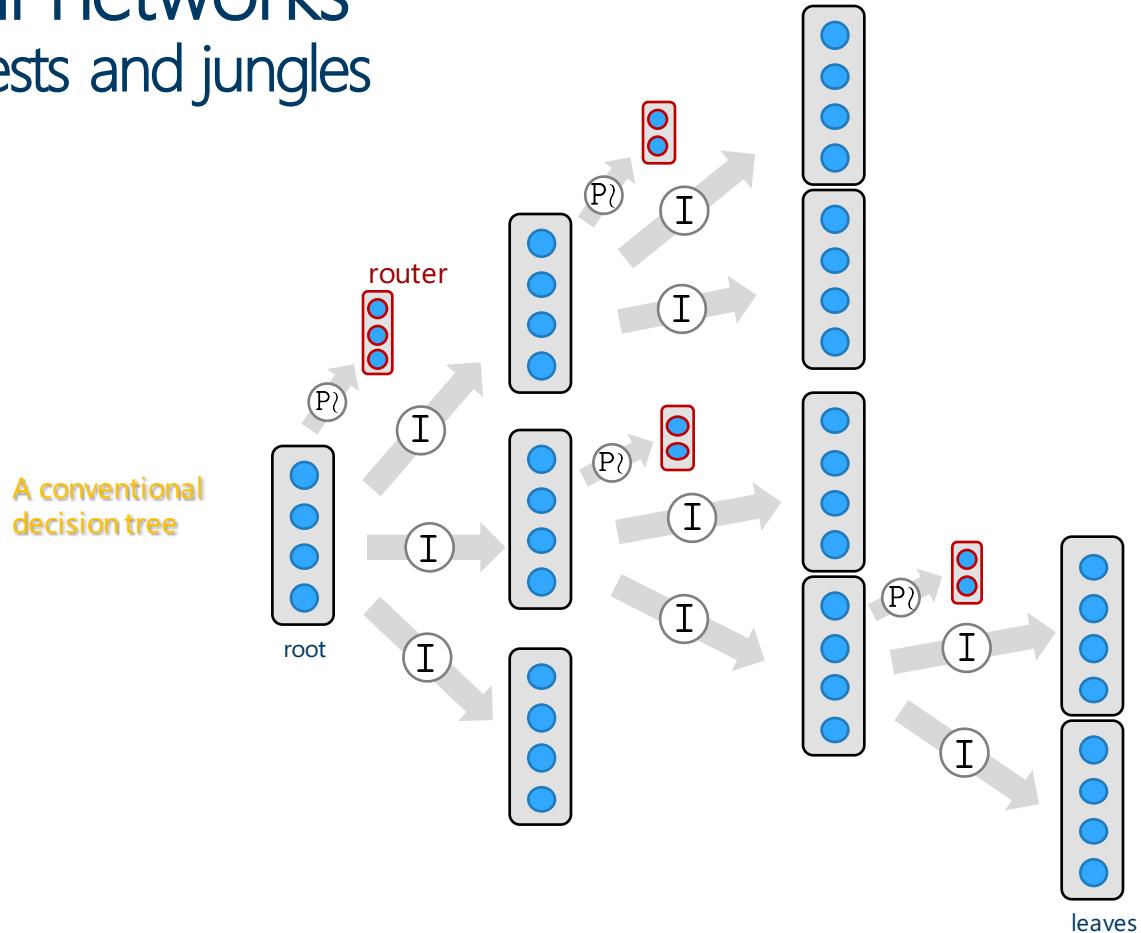
A conditional network



Conditional networks generalize (deep) networks



Conditional networks generalize forests and jungles

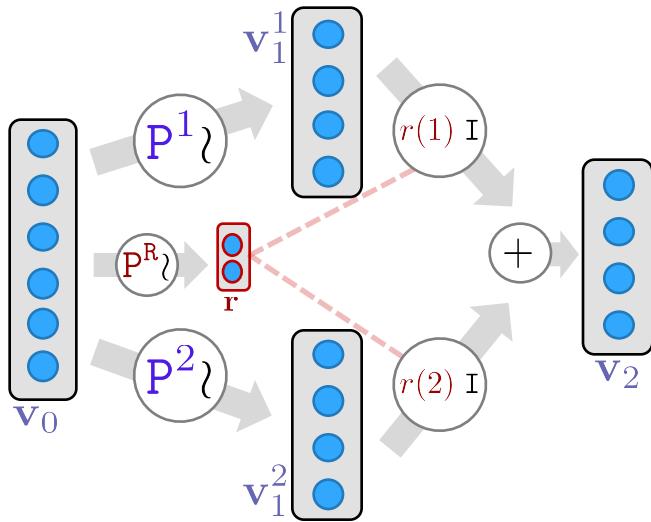


Outline

- **Introduction**
 - Trees, DAGs and deep neural networks (DNNs)
 - *Motivation:* Representing trees/DAGs as MLPs
 - *Motivation:* ReLUs and data routing
- **Conditional networks**
 - The model
 - **Training via back-propagation**
- **Experiments and results**
 - Conditional sparsification of a perceptron
 - Comparing architectures on ImageNet
 - Comparing architectures on CIFAR
 - Conditional ensembles
- **Conclusion**



Training via gradient-descent back-propagation



Valid in general for **soft routing weights** r .

All implemented in unmodified CAFFE (GPU/CPU).

$$\Rightarrow E(\theta) = \frac{1}{2} \sum_{\tau} (\mathbf{v}_2(\theta) - \mathbf{v}_2^*)^\top (\mathbf{v}_2(\theta) - \mathbf{v}_2^*) \quad \text{The training loss to be minimized.}$$

$\theta := \{P^R, \{P^j\}\}$ The parameters to optimize over

$\mathcal{T} = \{\dots, (\mathbf{v}_0, \mathbf{v}_2^*), \dots\}$ The labelled training set

$\mathbf{v}_2(\theta) = \mathbf{r}(\theta) \mathbf{V}_1(\theta)$ Network's forward mapping

$$\mathbf{V}_1 = \begin{bmatrix} \vdots & \vdots & \vdots \\ - & (\mathbf{v}_1^1)^\top & - \\ \vdots & \vdots & \vdots \end{bmatrix} \quad \text{Matrix of outputs for all routes}$$

$\mathbf{v}_1^j = \sigma(P^j \mathbf{v}_0)$ Intermediate output for j -th route

$\mathbf{r} = \sigma(P^R \mathbf{v}_0)$ Soft routing weights $\mathbf{r} \in \mathbb{R}^R$

$$\Delta\theta_{t+1} := -\rho \left. \frac{\partial E}{\partial \theta} \right|_t \quad \text{The parameter update rule}$$

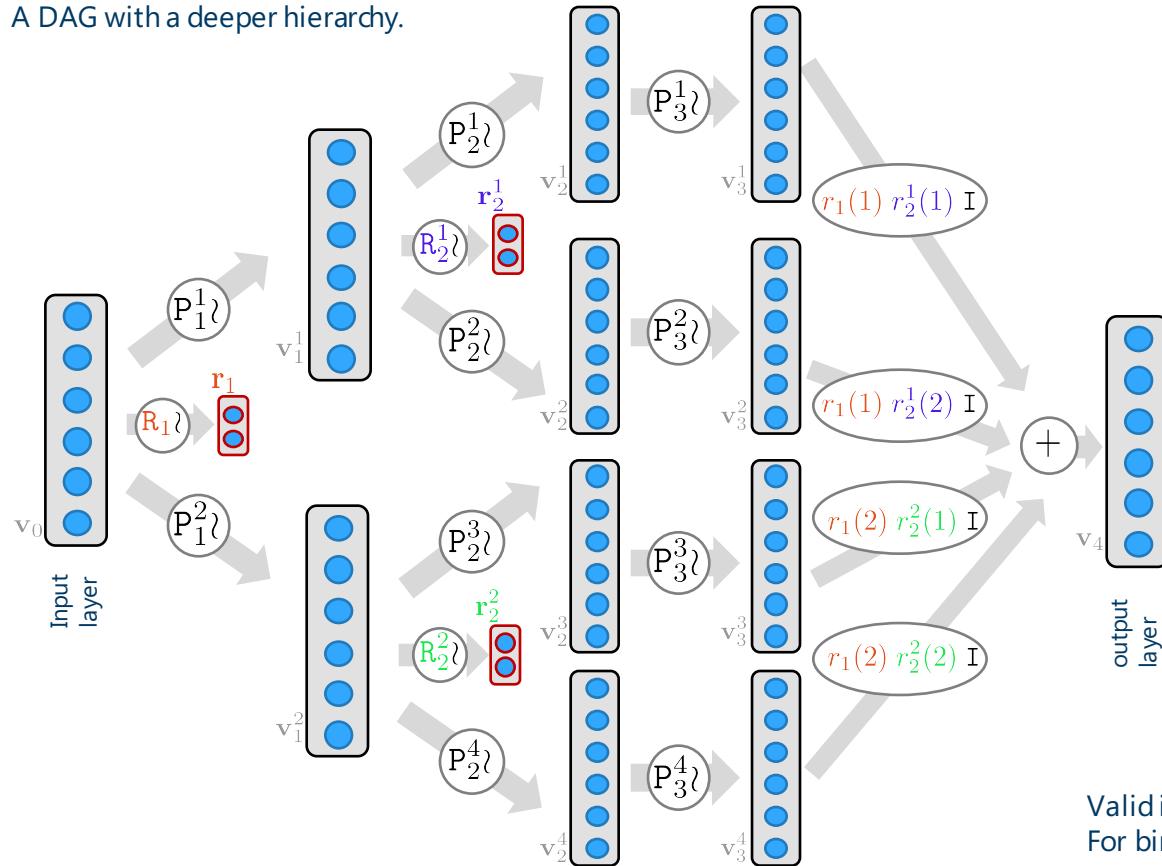
$$\Rightarrow \frac{\partial E}{\partial \theta} = \frac{\partial E}{\partial \mathbf{v}_2} \frac{\partial \mathbf{v}_2}{\partial \theta} = \frac{\partial E}{\partial \mathbf{v}_2} \left(\frac{\partial \mathbf{r}}{\partial P^R} \mathbf{V}_1 + \sum_j r(j) \frac{\partial \mathbf{v}_1^j}{\partial \phi^j} \frac{\partial \phi^j}{\partial P^j} \right) \quad \text{Chain rule}$$

$$\phi^j := P^j \mathbf{v}_0$$

Routing weights influence the back-propagating errors differently for different routes

Training a conditional network via back-propagation

A DAG with a deeper hierarchy.



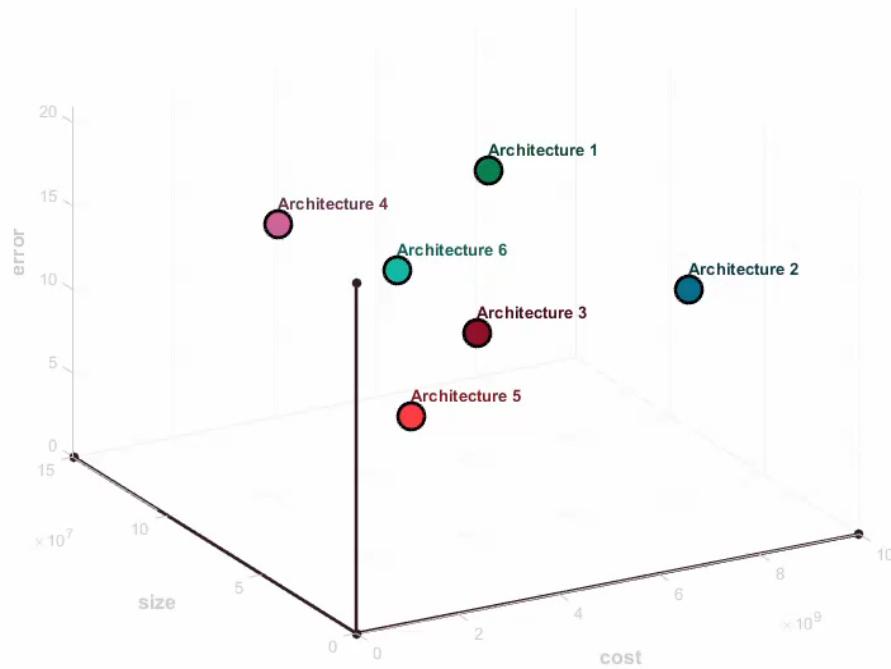
Valid in general for **soft routing weights r** .
For binary, hard weights the network structure simplifies.

Outline

- **Introduction**
 - Trees, DAGs and deep neural networks (DNNs)
 - *Motivation:* Representing trees/DAGs as MLPs
 - *Motivation:* ReLUs and data routing
- **Conditional networks**
 - The model
 - Training via back-propagation
- **Experiments and results**
 - Conditional sparsification of a perceptron
 - Comparing architectures on ImageNet
 - Comparing architectures on CIFAR
 - Conditional deep ensembles
- **Conclusion**



How to measure the efficacy of a DNN architecture ?



- Test accuracy
- Test-time compute cost
- Model size (num. params)
- ...
- Training-time compute cost
- Ease of parallelization
- Generalization to other datasets
- ...

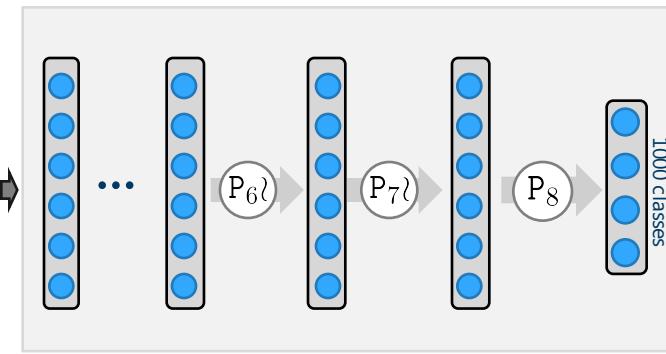
Outline

- **Introduction**
 - Trees, DAGs and deep neural networks (DNNs)
 - *Motivation:* Representing trees/DAGs as MLPs
 - *Motivation:* ReLUs and data routing
- **Conditional networks**
 - The model
 - Training via back-propagation
- **Experiments and results**
 - Conditional sparsification of a perceptron
 - Comparing architectures on ImageNet
 - Comparing architectures on CIFAR
 - Conditional ensembles
- **Conclusion**

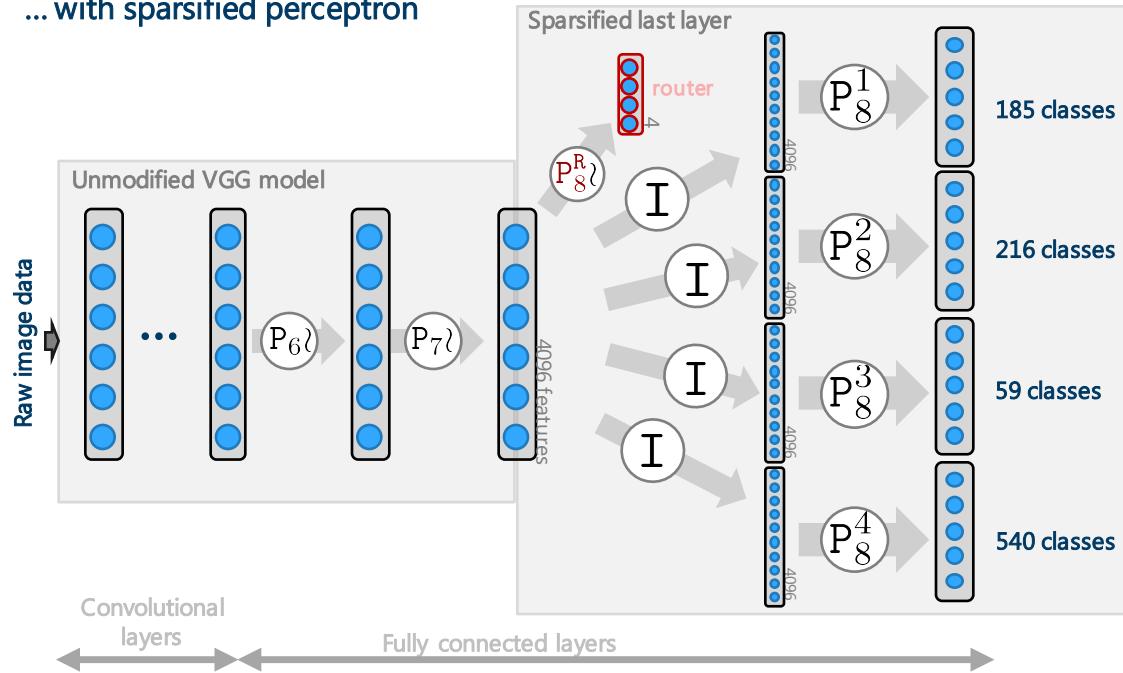


Conditional sparsification of a perceptron

Trained convolutional DNN

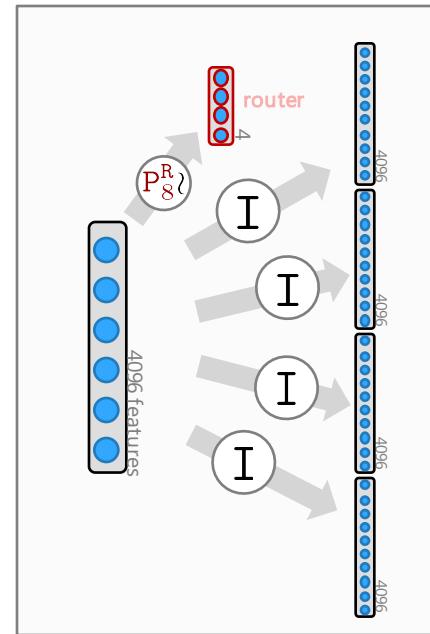
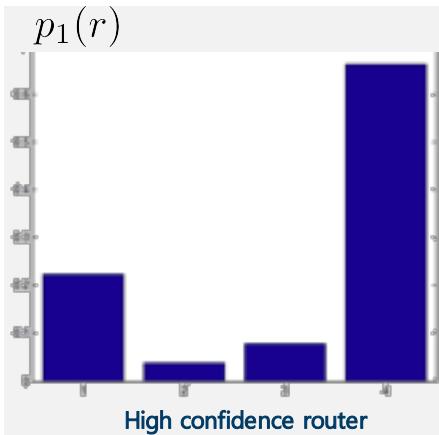


... with sparsified perceptron



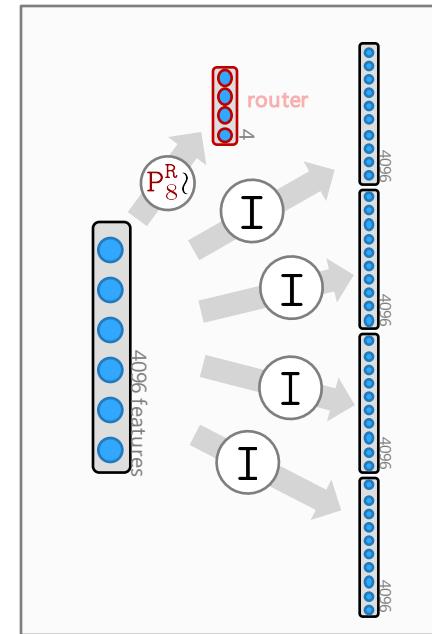
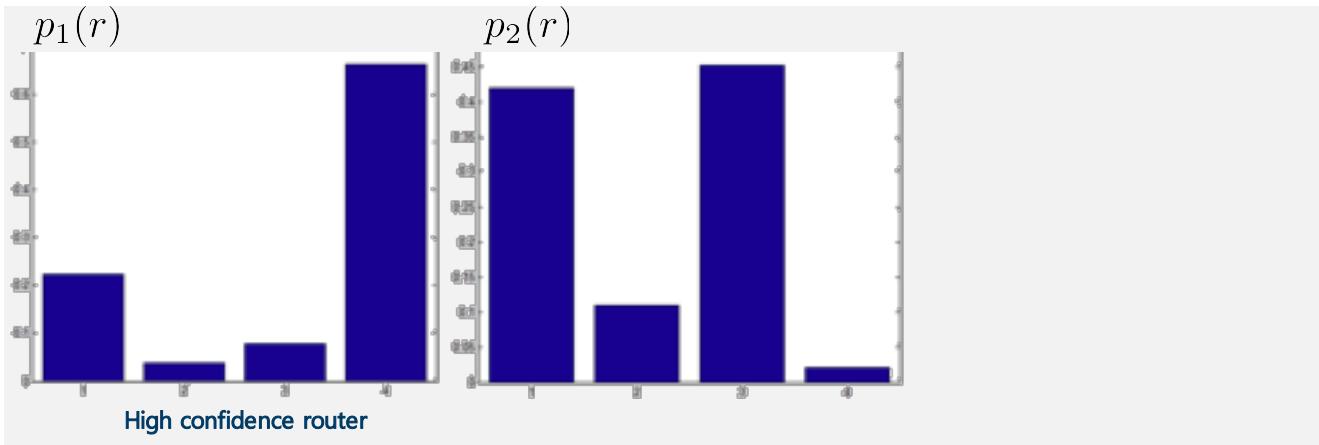
Conditional sparsification of a perceptron

Testing with multiple (hard) routes per input image



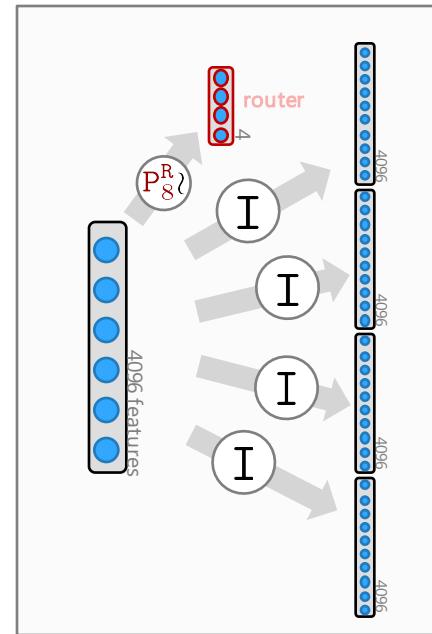
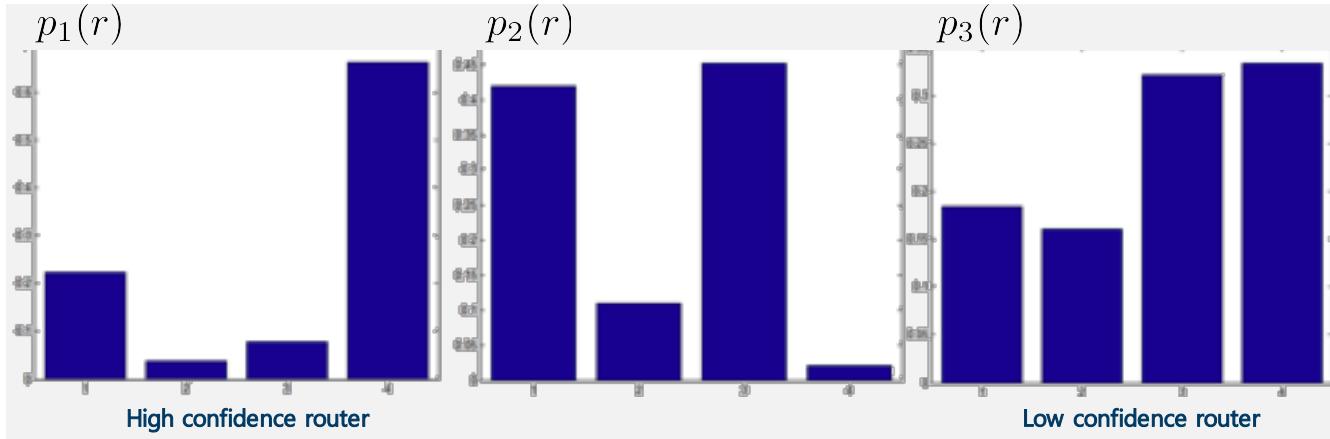
Conditional sparsification of a perceptron

Testing with multiple (hard) routes per input image



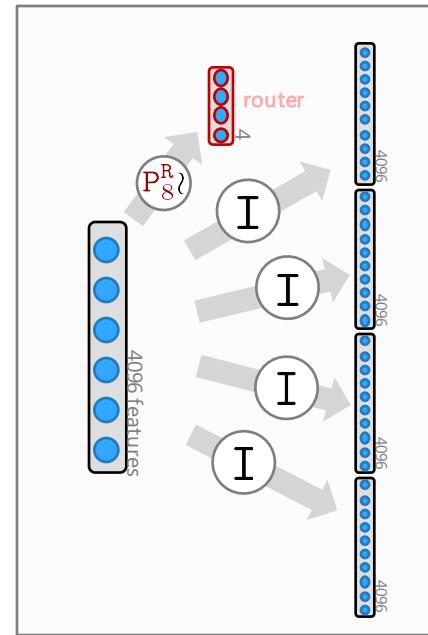
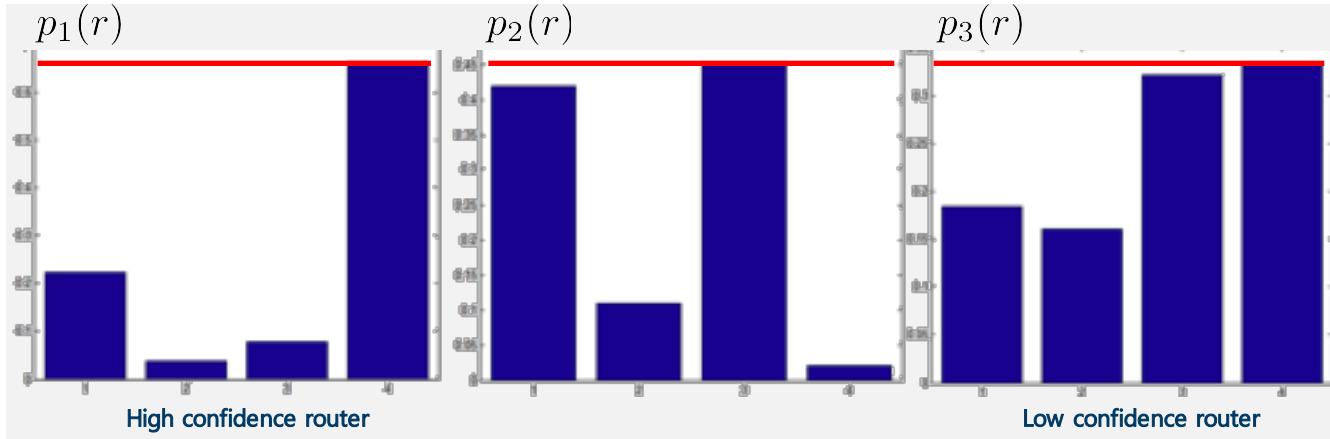
Conditional sparsification of a perceptron

Testing with multiple (hard) routes per input image



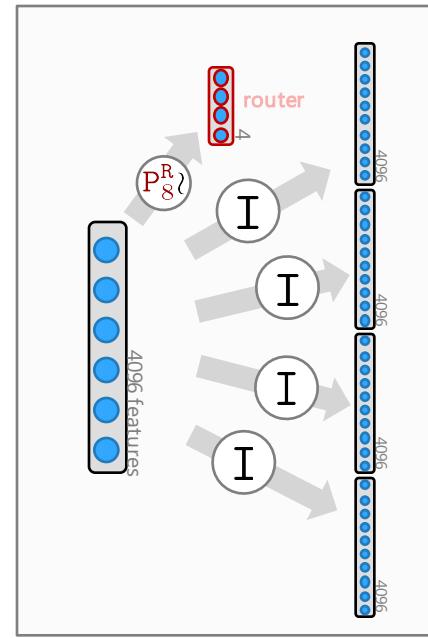
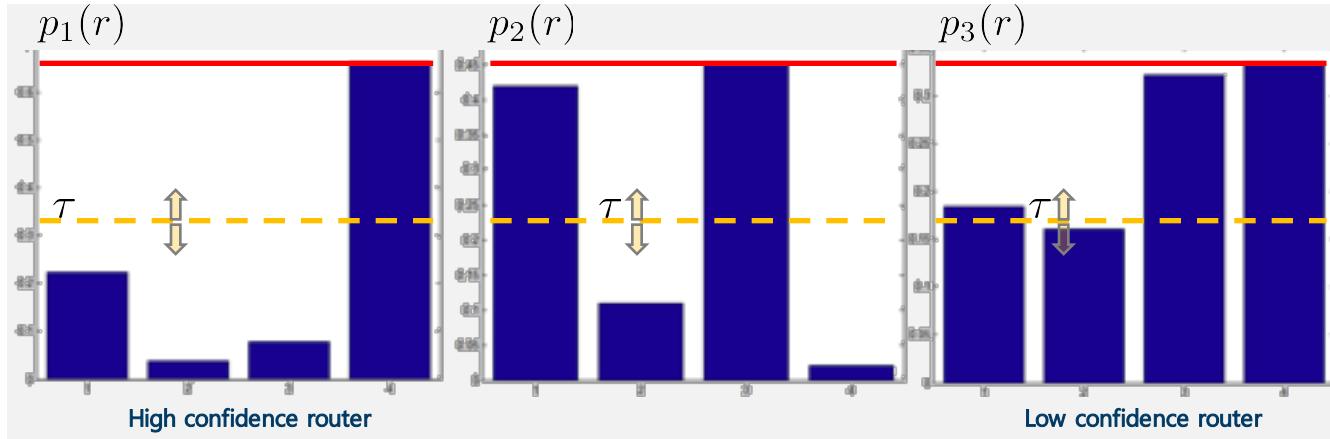
Conditional sparsification of a perceptron

Testing with multiple (hard) routes per input image



Conditional sparsification of a perceptron

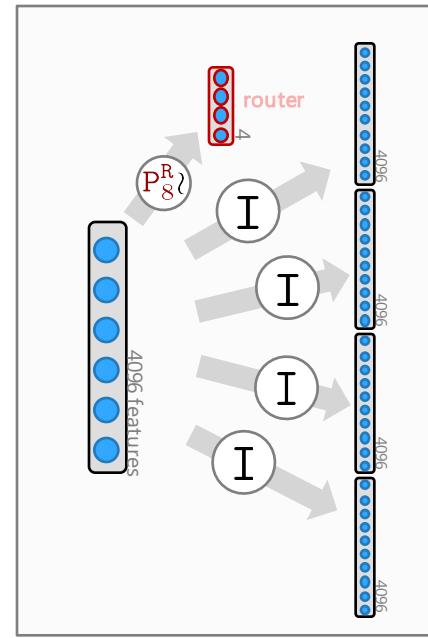
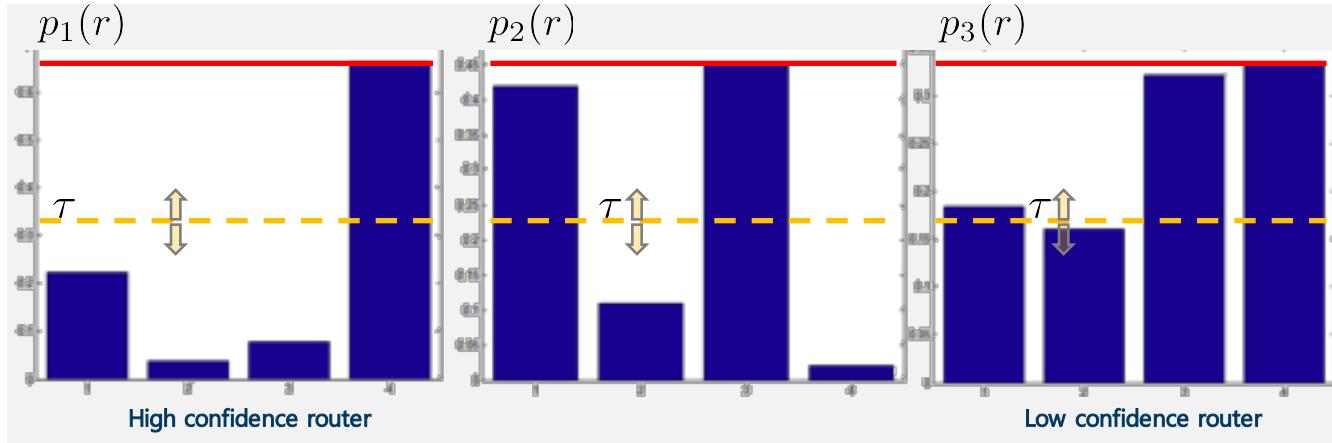
Testing with multiple (hard) routes per input image



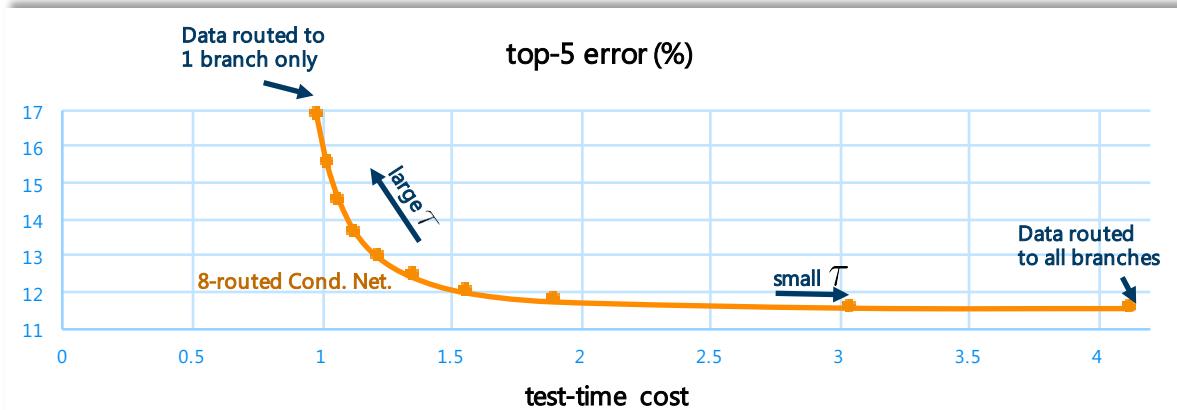
$$\tau = \alpha \max_r p(r) \quad \alpha \in [0, 1]$$

Conditional sparsification of a perceptron

Testing with multiple (hard) routes per input image



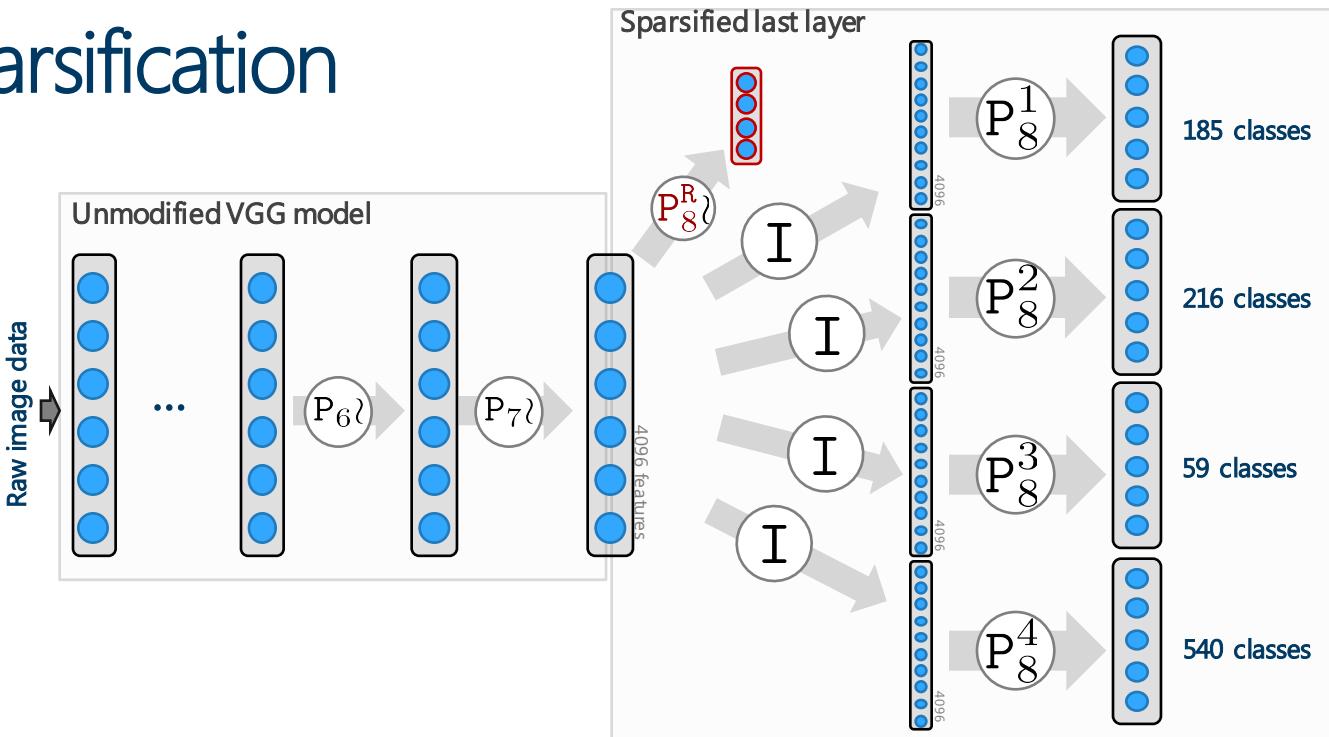
$$\tau = \alpha \max_r p(r) \quad \alpha \in [0, 1]$$



On-demand accuracy-compute trade-off

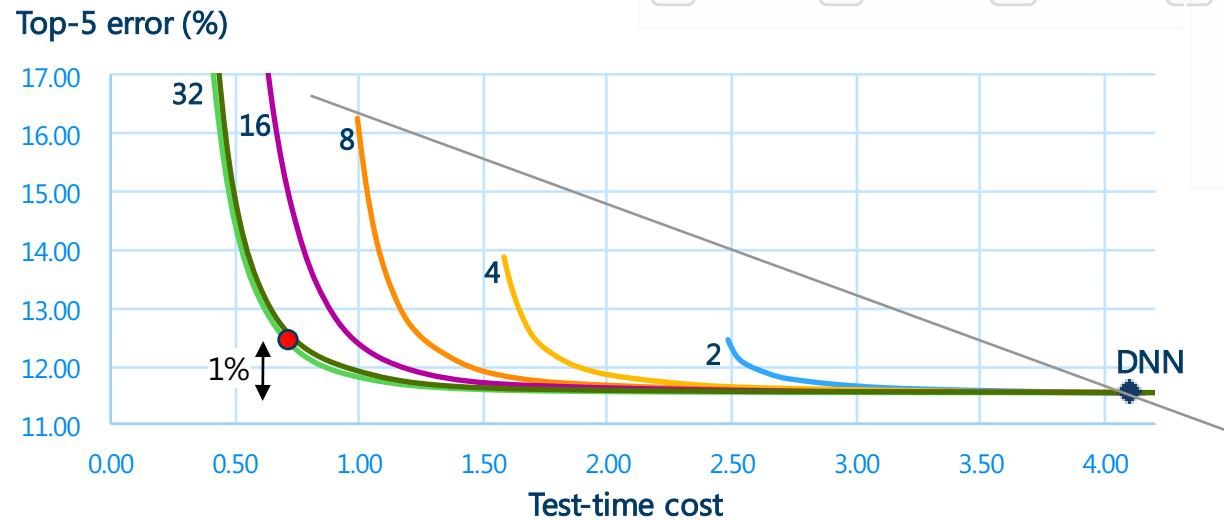
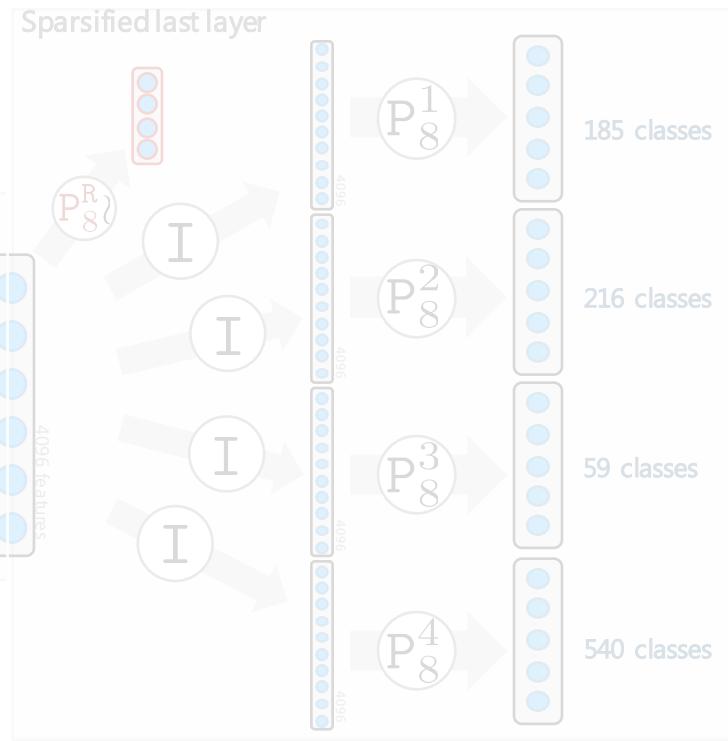
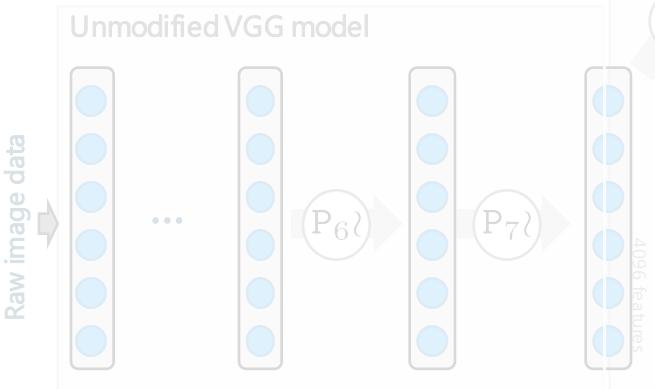
Conditional sparsification

On-demand accuracy-compute
trade-off



Conditional sparsification

On-demand accuracy-compute trade-off



- For 1% increase of T5E:
 - ~ X 6 speed up
 - ~ same model size

Conditional sparsification: learned super classes for a 16-ary CONNET

spC 1 (55) spC 2 (77) spC 3 (48) spC 4 (120) spC 5 (10) spC 6 (185) spC 7 (59) spC 8 (41) spC 9 (86) spC 10 (65) spC 11 (50) spC 12 (60) spC 13 (41) spC 14 (16) spC 15 (41) spC 16 (46)

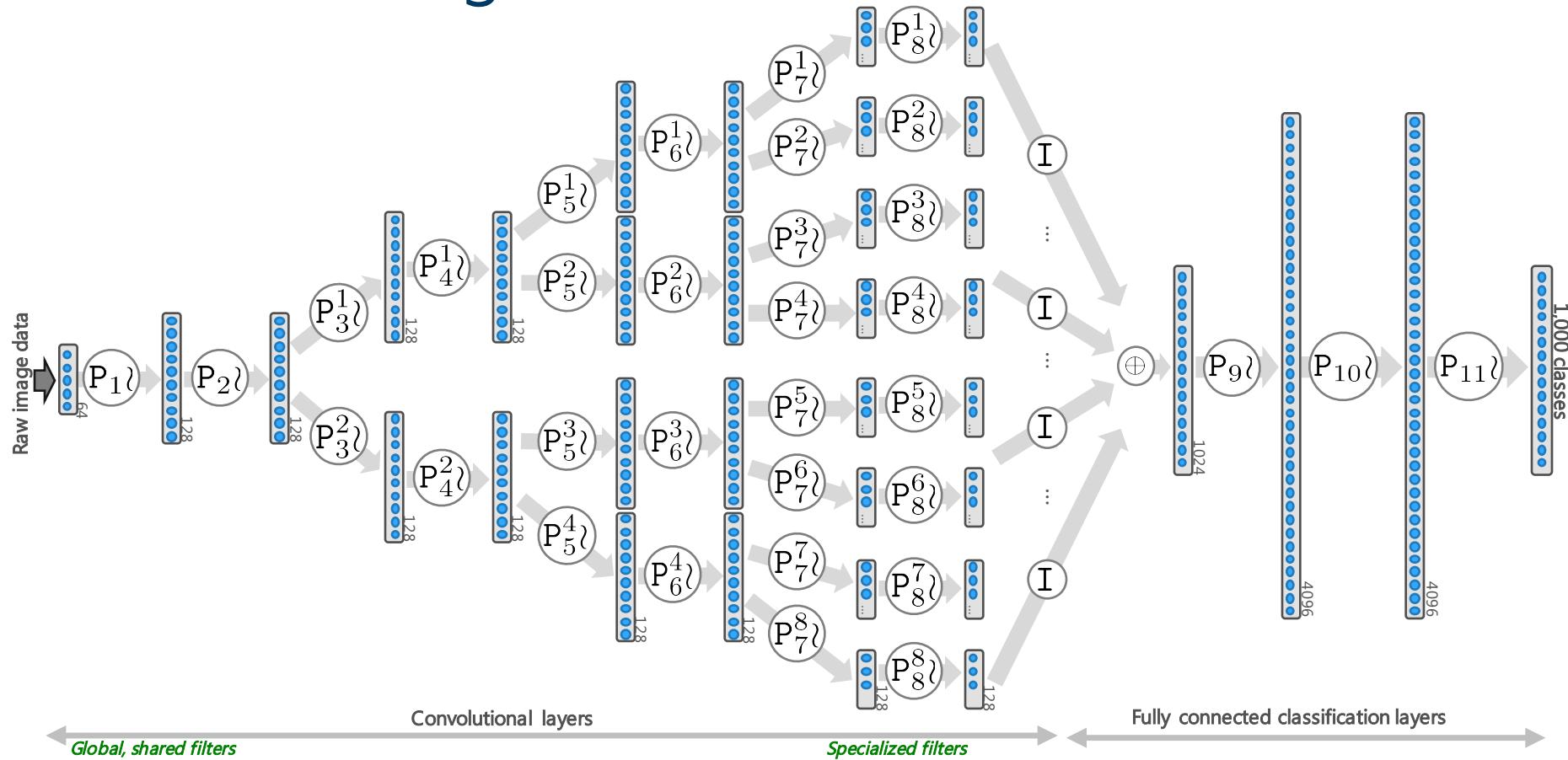
vault	pinwheel	hip	tibetan terrier	electric fan	american crow	trolleybus	projectile	meat loaf	oxcart	stage	scuba diver	armadillo	giant panda
upright	pillow	yellow lady	scotch terrier	dumbbell	european goldfinch	trailer truck	pole	dough	african elephant	soccer ball	coral reef	slug	lesser panda
turnstile	pajama	daisy	standard scottish terrier	doormat	limpkin	tractor	planetarium	chocolate sauce	indian elephant	snorkel	sidewinder	indri	indri
tub	oxygen mask	cardoon	giant schnauzer	disk brake	crane	tow truck	pirate	carbonara	llama	ski	loggerhead sea turtle	madagascar diamondback	madagascar
throne	overskirt	spider web	miniature schnauzer	digital water	bittern	thresher	pier	pomegranate	arabian camel	shopping cart	puffer	squid	squid
theater curtain	neck brace	lycaenid	boston bull terrier	digital clover	american egret	tank	picket fence	custard apple	gazelle	saxophone	lionfish	horned viper	horned viper
table lamp	mortarboard	sulphur butterfly	dandie dinmont terrier	dial telephone	little blue penguin	park bench	pedestal	jackfruit	impala	rugby ball	gar	green mamba	green mamba
studio couch	mitten	cabbage butterfly	australian cattle dog	desktop computer	flamingo	streetcar	parachute	banana	hartebeest	racket	sturgeon	indian cobra	howler monkey
stove	miniskirt	monarch	cairn terrier	crock pot	spoonbill	steam locomotive	palace	pineapple	ibex	puck	ping-pong ball	rock python	capuchin
sliding door	military uniform	ringlet	airedale terrier	crash helmet	black stork	sports car	paddlewheel	fig	bighorn sheep	parallel bars	anemone fish	boa constrictor	marmoset
shower curtain	mask	admiral	sealyham terrier	corkscrew	white stork	snowplow	obelisk	ram	panpipe	rock beauty	web site	night snake	proboscis monkey
shoji	maillot	damselfly	lakeland terrier	computer keyboard	black swan	obligation	mountain goat	bison	paddle	pancake	vine snake	vine snake	colobus
rocking chair	maillot	dragonfly	wire-haired yorkshire terrier	combination	goose	school bus	mosque	strawberry	water buffalo	coho salmon	sturgeon	langur	langur
restaurant	mailbag	lacewing	norwich terrier	coil	red-breasted merganser	recreational vehicle	monastery	granny smelt	ox	eel	anemone fish	indian cobra	howler monkey
refrigerator	loafer	leafhopper	norfolk terrier	coffee pot	drake	recreational vehicle	monastery	mushroom	hippopotamus	baracouta	rock python	rock python	capuchin
radiator	lab coat	cicada	norfolk terrier	cocktail shaker	toucan	racer	mobile home	bell pepper	warthog	sea cucumber	boa constrictor	boa constrictor	marmoset
quilt	knee pad	mantis	irish terrier	cleaver	jacamar	missile	home bell pepper	artichoke	wild boar	toyshop	night snake	night snake	proboscis monkey
prison	kimono	cockroach	kerry blue terrier	chest	hummingbird	missile	mobile home	cucumber	moped	garter snake	vine snake	vine snake	colobus
pool table	jersey	jersey	border terrier	chain	hummingbird	missile	mobile home	hog	maypole	king snake	macaque	macaque	langur
plate rack	jean	walking stick insect	bedlington terrier	cellular telephone	bee eater	missile	mobile home	marimba	starfish	sea urchin	baboon	baboon	langur
patio	hoop skirt	cricket	american staffordshire terrier	cd player	coucal	missile	mobile home	zebra	tobacco shop	toyshop	macaque	macaque	langur
organ	handkerchief	grasshopper	staffordshire bull terrier	cassette player	lorikeet	missile	mobile home	zebra	sea lion	toyshop	macaque	macaque	langur
mosquito net	gown	ant	weimaraner	cassette tape	sulphur-crested cockatoo	missile	mobile home	sea lion	dugong	toyshop	macaque	macaque	langur
microwave	gas mask	bee	scottish deerhound	cash machine	macaw	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
medicine chest	fur coat	fly	saluki	carton	african grey parrot	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
library	feather boa	weevil	otterhound	carpenter's	partridge	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
lampshade	dishrag	rhinoceros beetle	norwegian elkhound	car mirror	quail	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
home theater	diaper	dung beetle	ibiza hound	can opener	peacock	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
guillotine	cuirass	leaf beetle	whippet	candle	prairie chicken	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
grand piano	cowboy hat	cowboy boot	italian greyhound	caldron	ruffed grouse	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
four-poster bed	cowboy boot	long-horned beetle	irish wolfhound	buckle	ptarmigan	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
fold-down sofa	clog	ground beetle	leaf beetle	bucket	black grouse	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
fire screen	clothing	ladybug	tiger beetle	great tit	great tit	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
file cabinet	clip hasp	isopod	redback spider	broccoli	blue tit	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
entertainment center	chipboard	isopod	endemic fox	brussels sprouts	blue tit	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
dishwasher	carabiner	lobster crab	water hound	hollyhock	bullock's kite	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
dinghy	bulletproof vest	crayfish	black-and-tan hound	hollyhock	binoculars	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
table	breastplate	fiddler crab	bloodhound	hollyhock	binder	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
crib	brassiere	rock crab	beagle	hollyhock	beer bottle	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
cradle	bow tie	snail	basset hound	hollyhock	magpie	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
	bonnet	centipede	goblet	hollyhock	jay	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
	tick	afghan hound	goblet	hollyhock	bulbul	missile	mobile home	sea lion	ringneck pheasant	toyshop	macaque	macaque	langur
furniture		clothes		insects		dogs		drinking		tools		birds	
												outdoor	
												large animals	
												music/sports	
												fruits	
												shops	
												reptiles	
												small animals	

Outline

- **Introduction**
 - Trees, DAGs and deep neural networks (DNNs)
 - *Motivation:* Representing trees/DAGs as MLPs
 - *Motivation:* ReLUs and data routing
- **Conditional networks**
 - The model
 - Training via back-propagation
- **Experiments and results**
 - Conditional sparsification of a perceptron
 - **Comparing architectures on ImageNet**
 - Comparing architectures on CIFAR
 - Conditional ensembles
- **Conclusion**

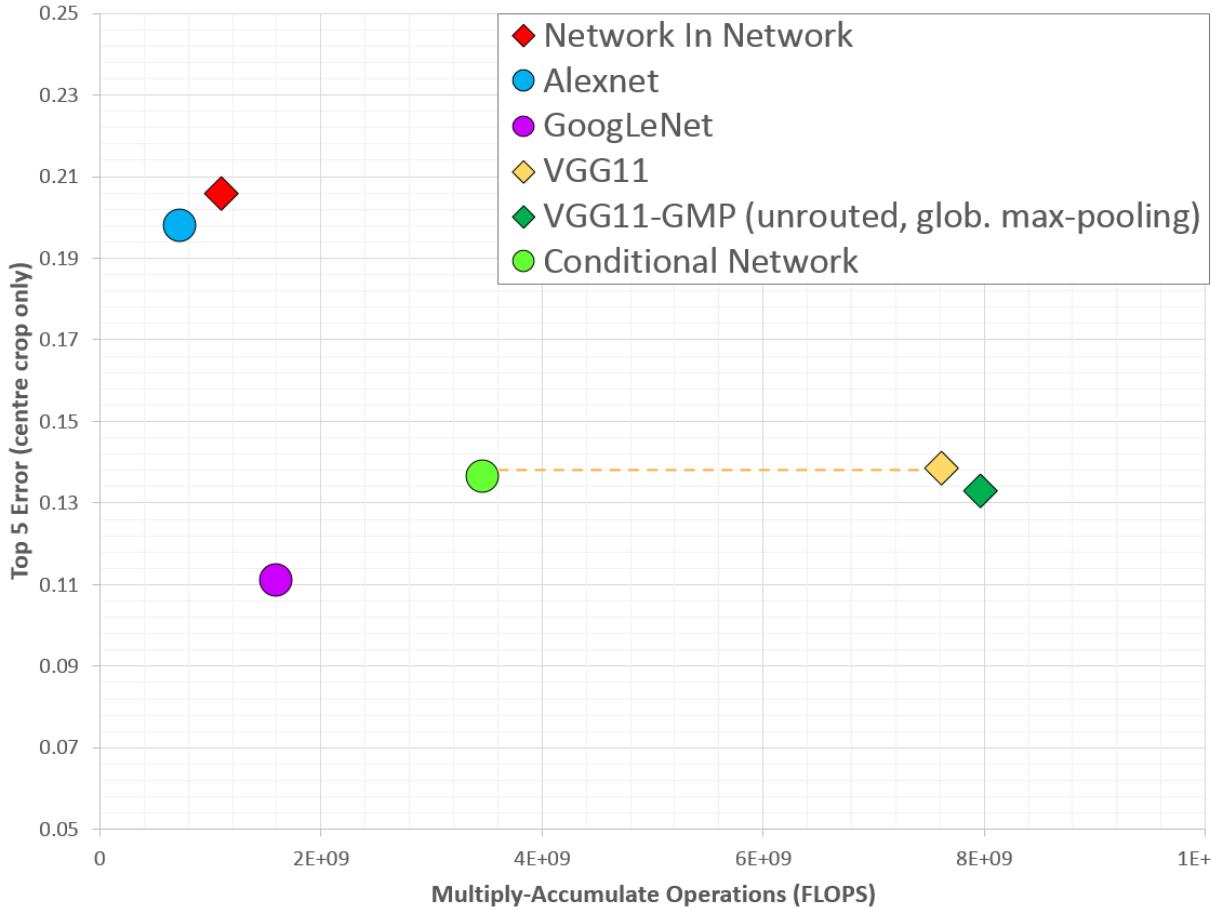


Results on ImageNet-1000



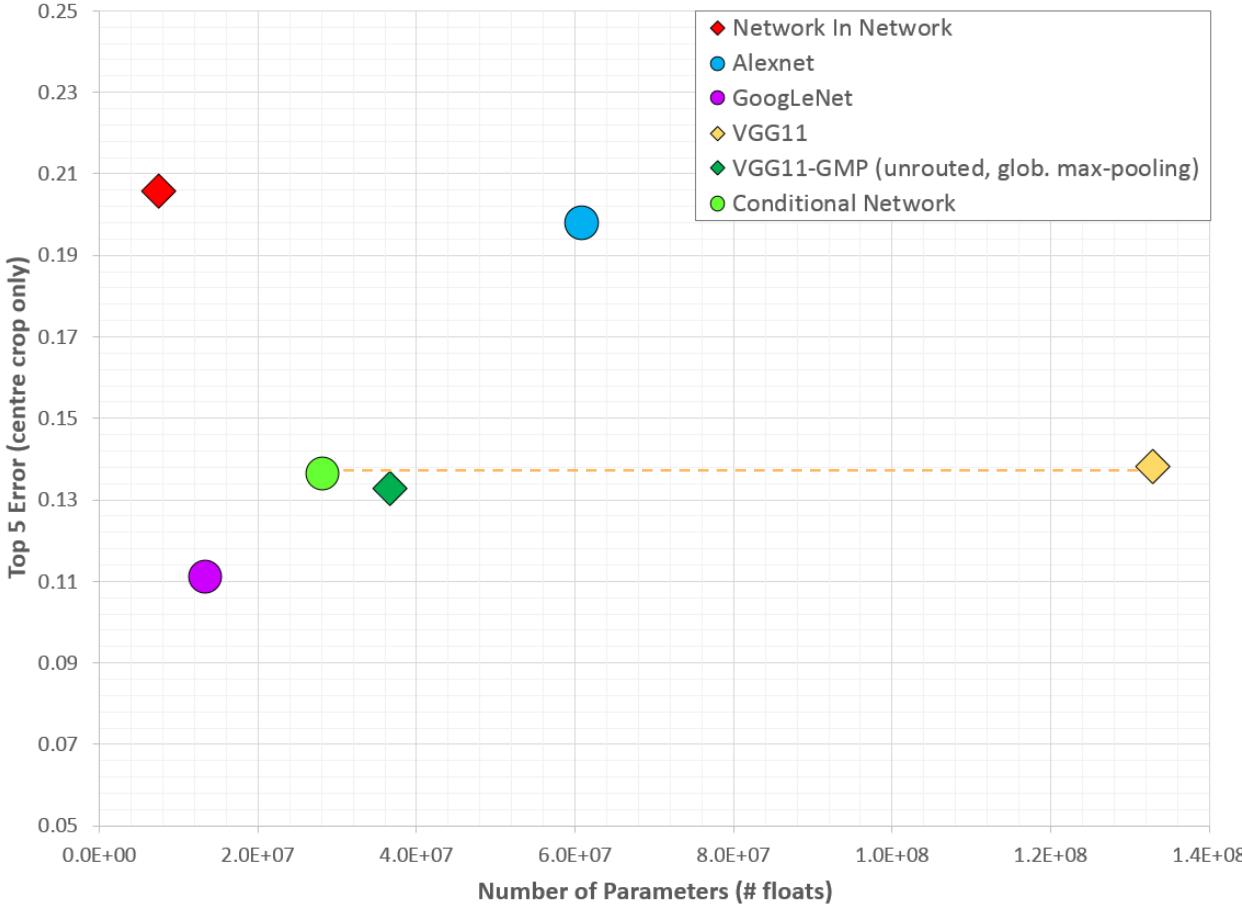
Results on ImageNet-1000

accuracy vs compute

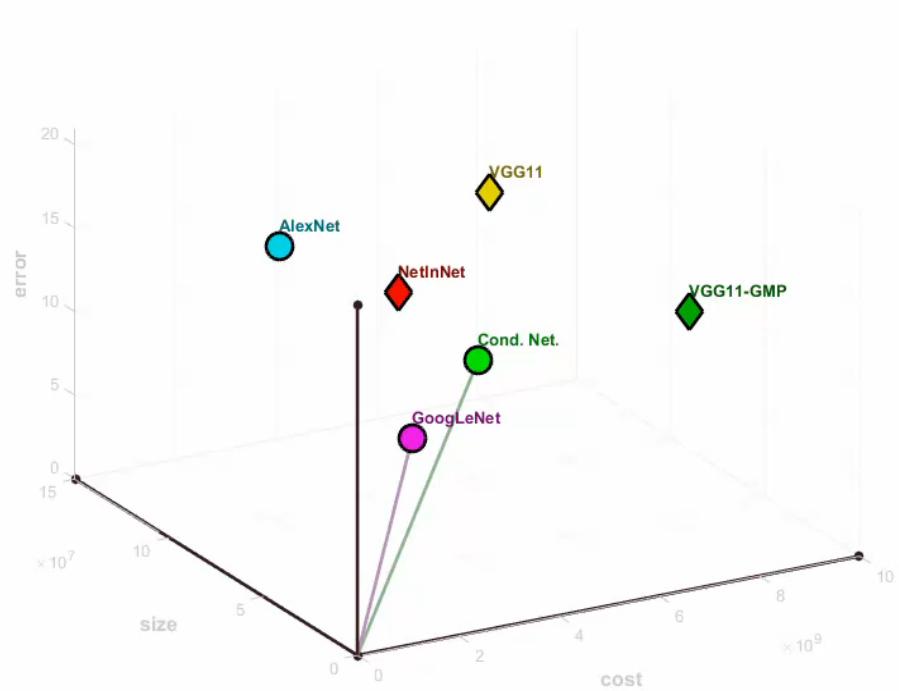


Results on ImageNet-1000

accuracy vs model size



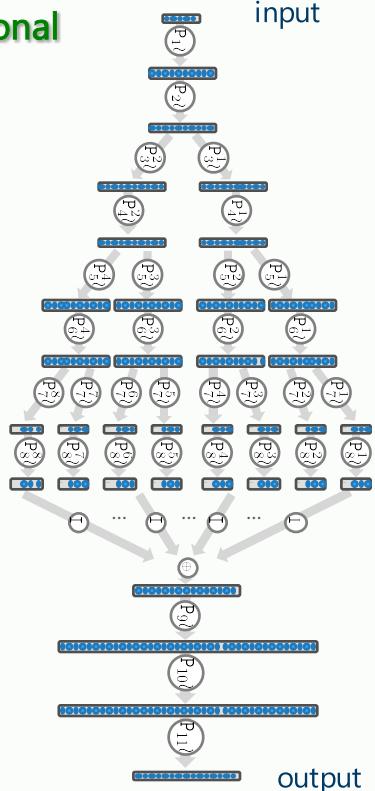
Results on ImageNet-1000



- Top-5 error
- Test-time compute cost (num. mult - add ops)
- Model size (num. params)

Results on ImageNet-1000

Our conditional network



Implicitly-routed conditional networks

Simpler architecture

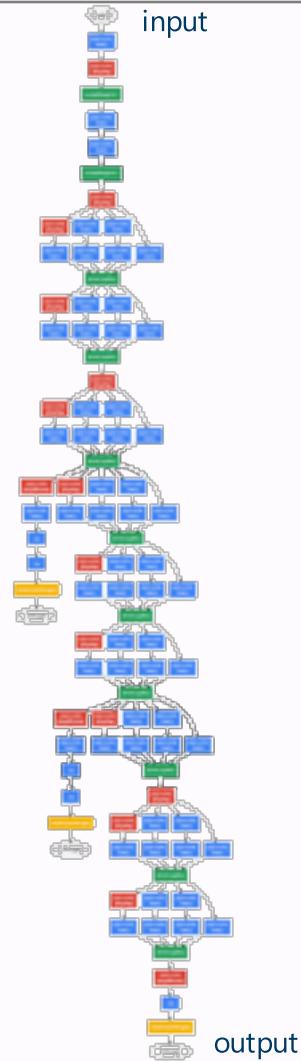
Intermediate, manually-weighted losses

Easier to train

Low-dimension embeddings

GoogLeNet

22 layers (~100 blocks)



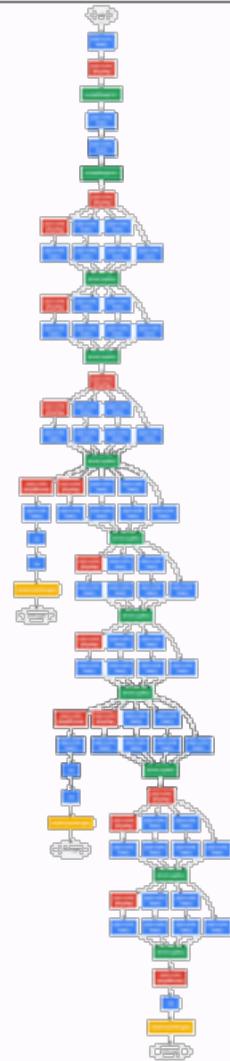
Results on ImageNet-1000

6 Training Methodology

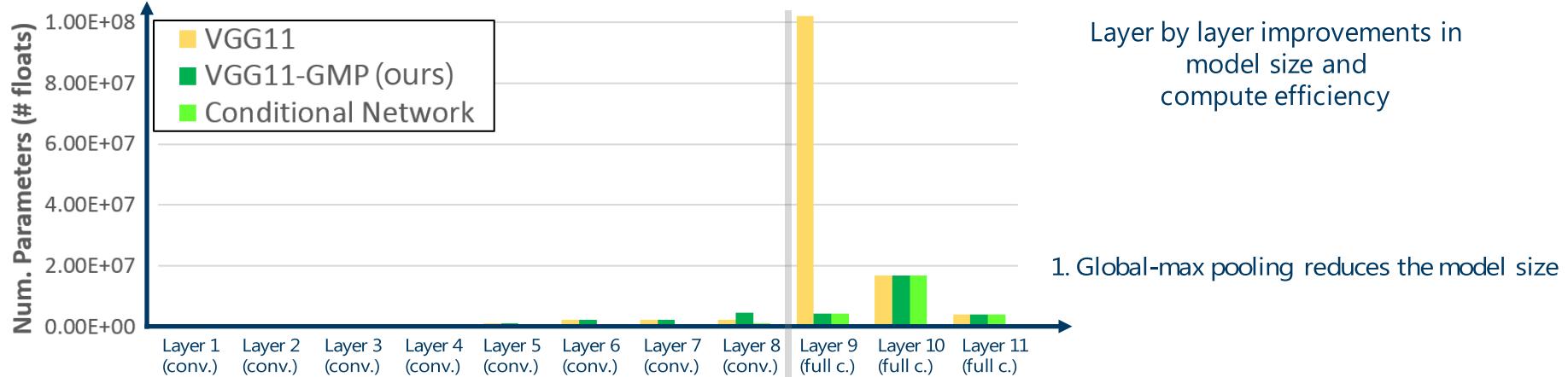
Our networks were trained using the DistBelief [4] distributed machine learning system using modest amount of model and data-parallelism. Although we used CPU based implementation only, a rough estimate suggests that the GoogLeNet network could be trained to convergence using few high-end GPUs within a week, the main limitation being the memory usage. Our training used asynchronous stochastic gradient descent with 0.9 momentum [17], fixed learning rate schedule (decreasing the learning rate by 4% every 8 epochs). Polyak averaging [13] was used to create the final model used at inference time.

Our image sampling methods have changed substantially over the months leading to the competition, and already converged models were trained on with other options, sometimes in conjunction with changed hyperparameters, like dropout and learning rate, so it is hard to give a definitive guidance to the most effective single way to train these networks. To complicate matters further, some of the models were mainly trained on smaller relative crops, others on larger ones, inspired by [8]. Still, one prescription that was verified to work very well after the competition includes sampling of various sized patches of the image whose size is distributed evenly between 8% and 100% of the image area and whose aspect ratio is chosen randomly between 3/4 and 4/3. Also, we found that the photometric distortions by Andrew Howard [8] were useful to combat overfitting to some extent. In addition, we started to use random interpolation methods (bilinear, area, nearest neighbor and cubic, with equal probability) for resizing relatively late and in conjunction with other hyperparameter changes, so we could not tell definitely whether the final results were affected positively by their use.

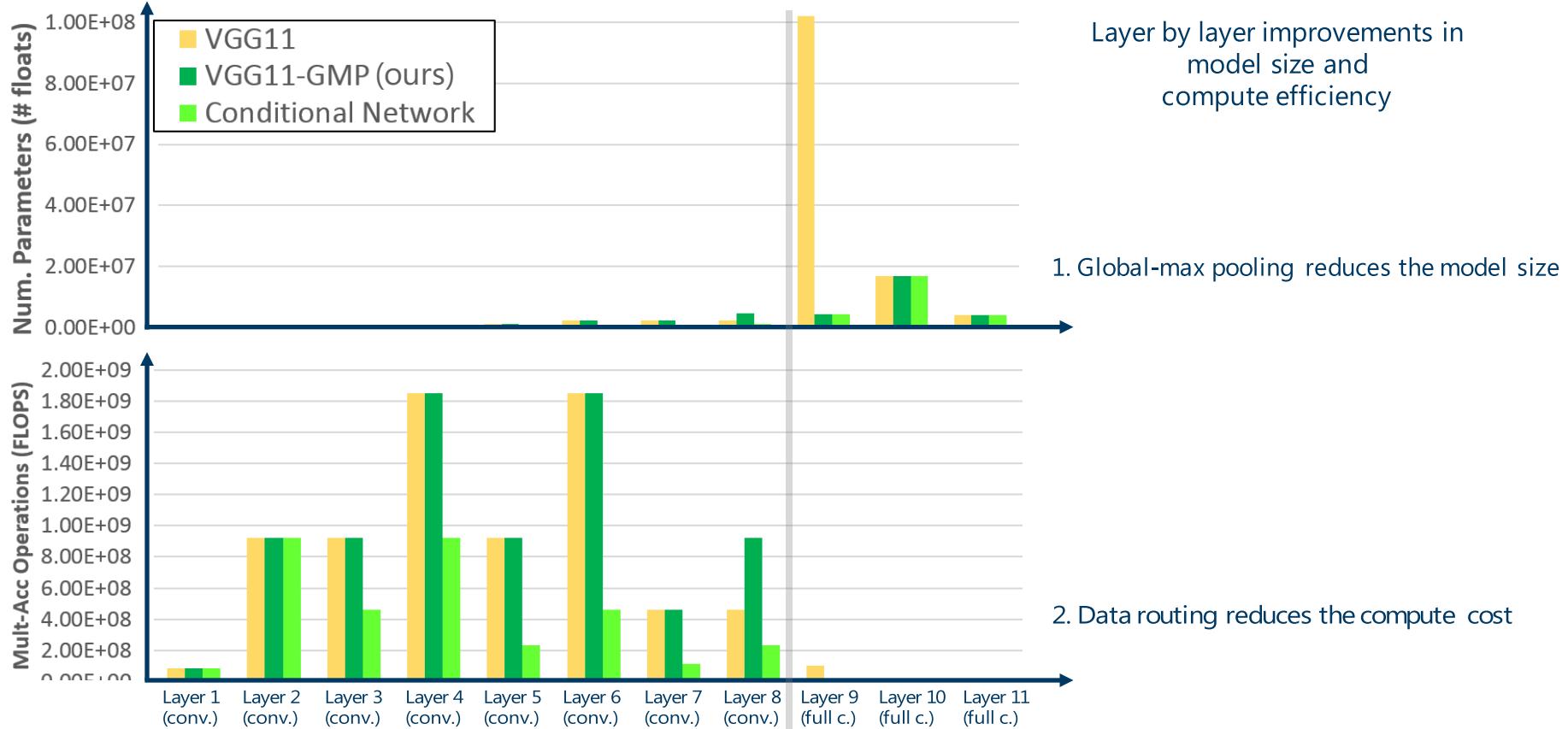
GoogLeNet



Results on ImageNet-1000



Results on ImageNet-1000

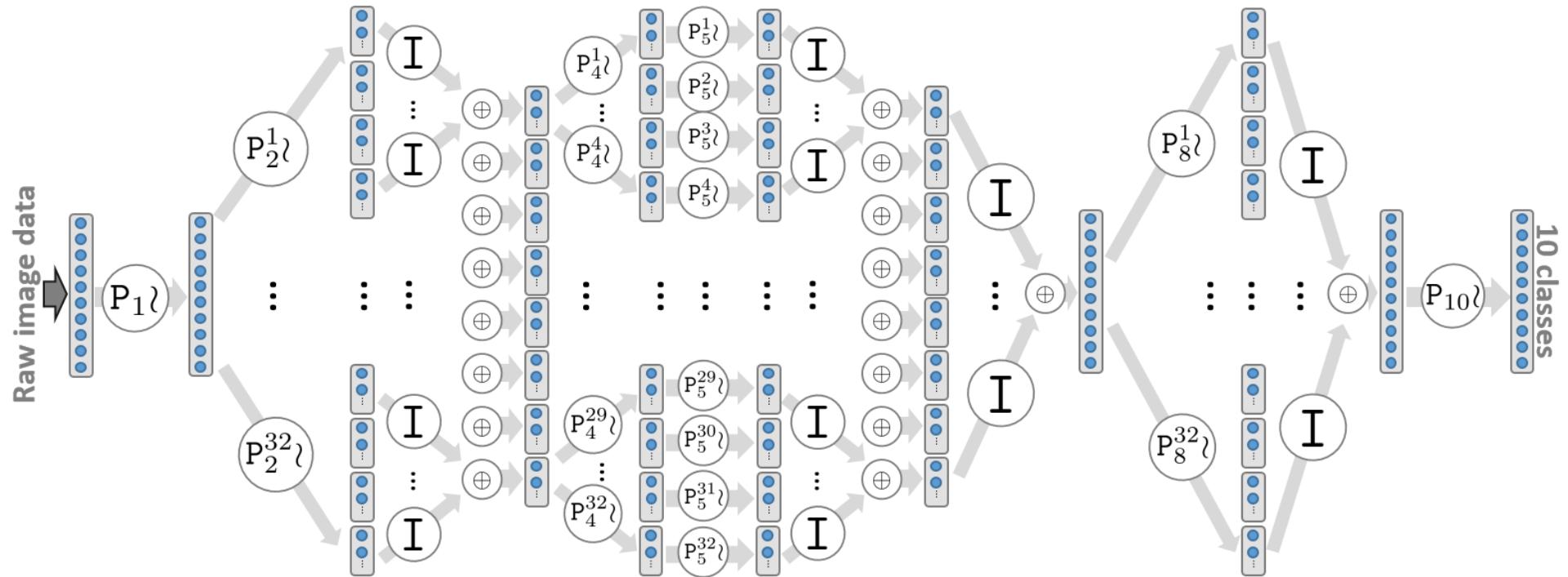


Outline

- **Introduction**
 - Trees, DAGs and deep neural networks (DNNs)
 - *Motivation:* Representing trees/DAGs as MLPs
 - *Motivation:* ReLUs and data routing
- **Conditional networks**
 - The model
 - Training via back-propagation
- **Experiments and results**
 - Conditional sparsification of a perceptron
 - Comparing architectures on ImageNet
 - **Comparing architectures on CIFAR**
 - Conditional ensembles
- **Conclusion**



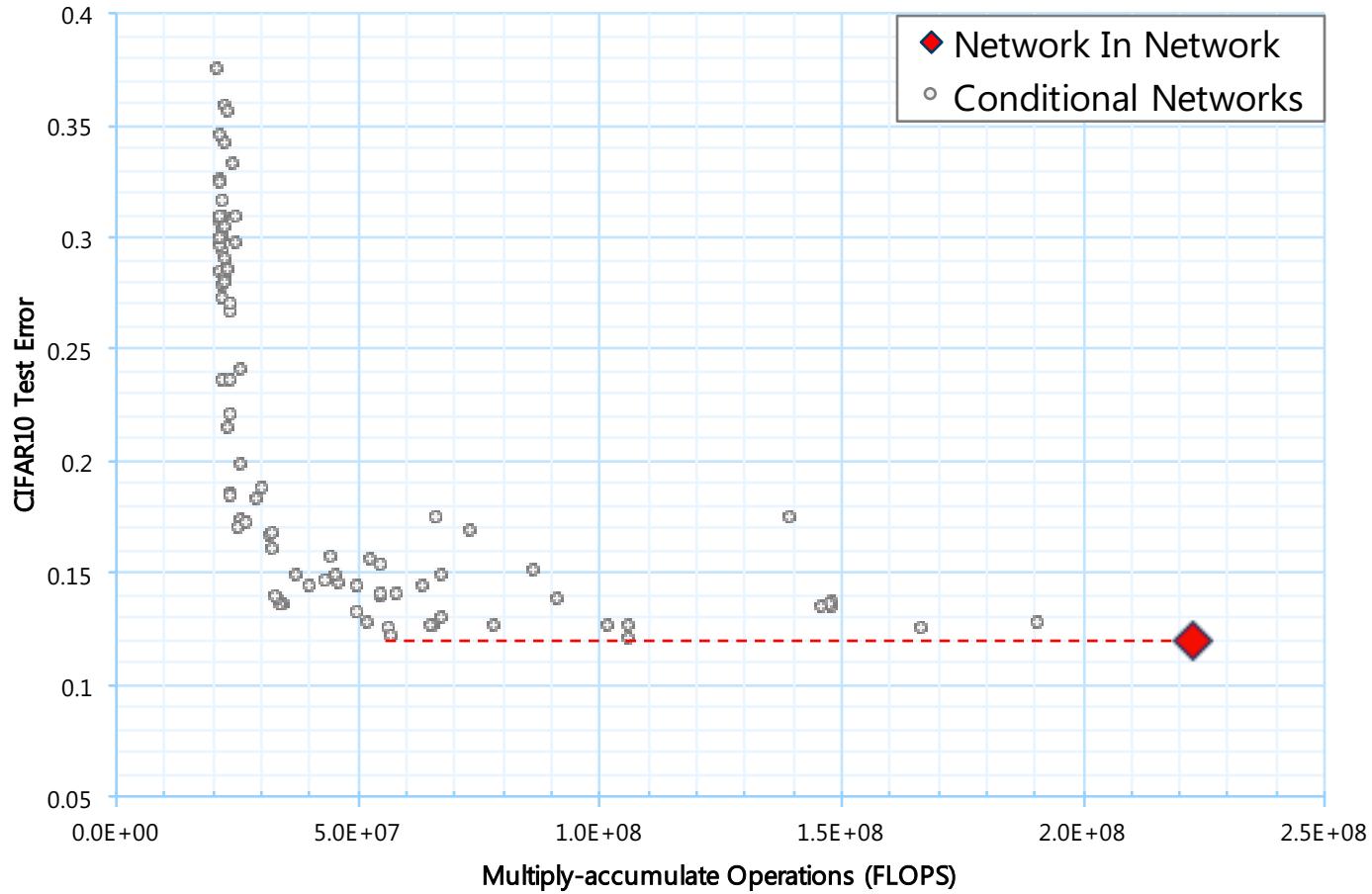
Results on CIFAR-10



Network parameters and architecture have been learned via *Bayesian optimization* (<https://www.whetlab.com/>)

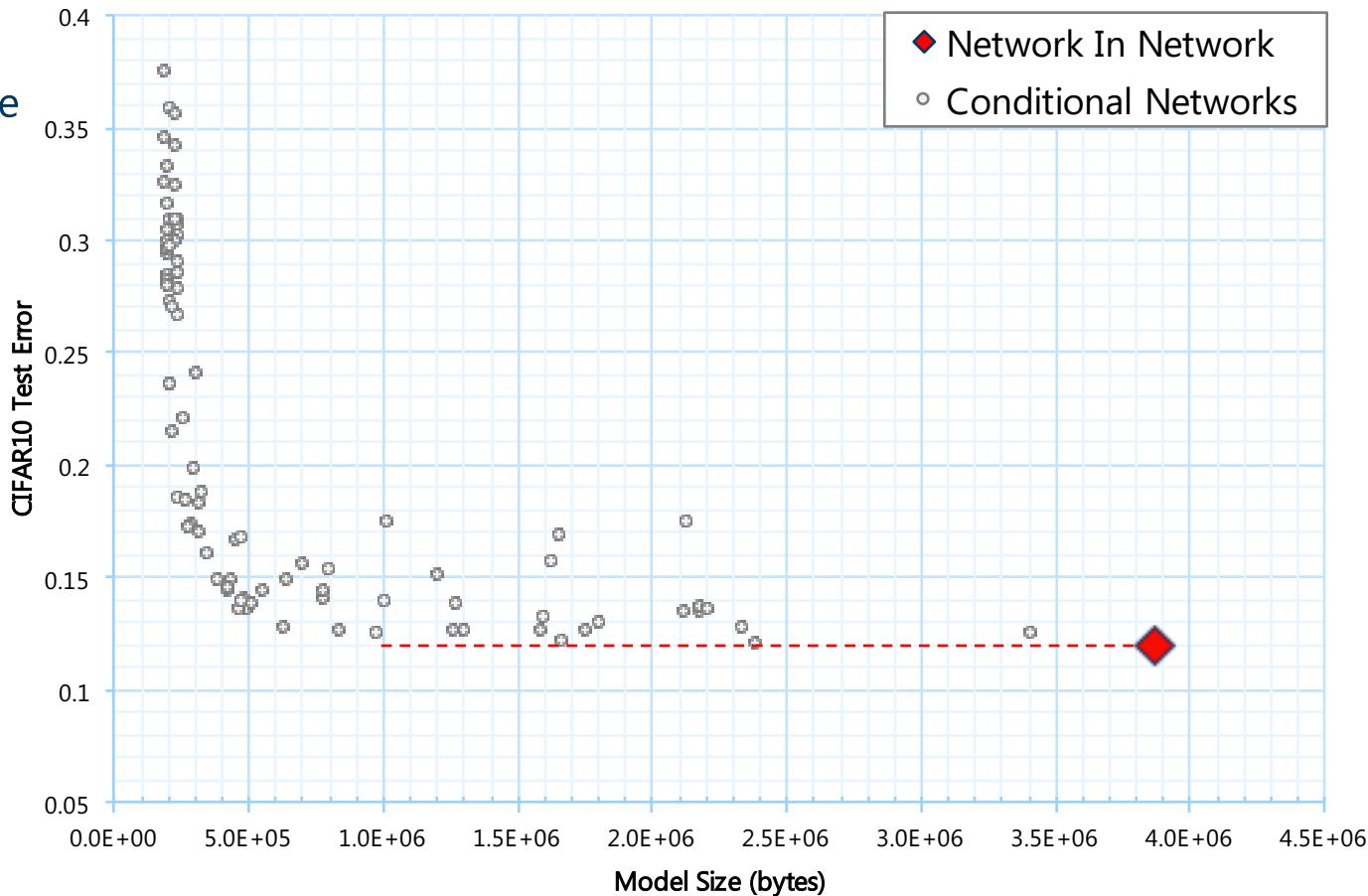
Results on CIFAR-10

accuracy vs compute

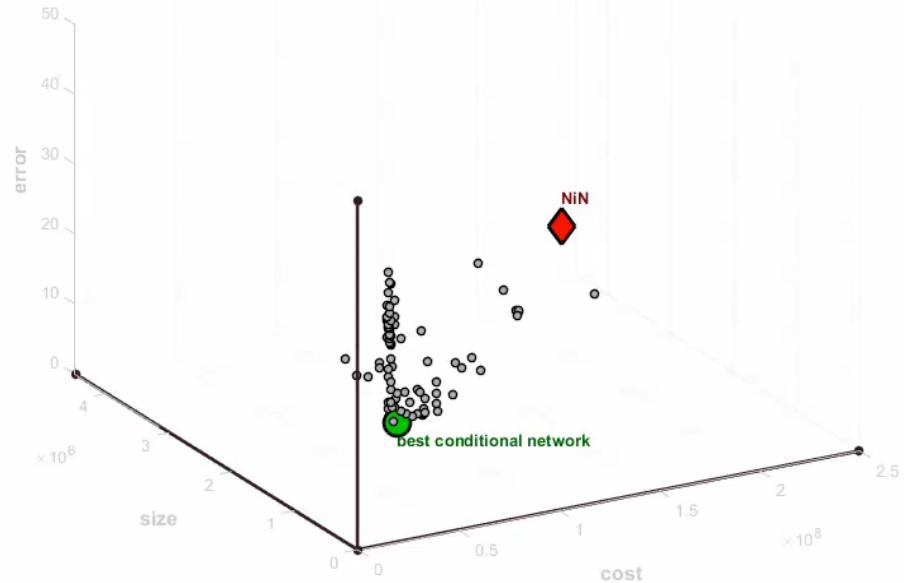


Results on CIFAR-10

accuracy vs model size



Results on CIFAR-10



- Top-1 error
- Test-time compute cost (num. mult - add ops)
- Model size (bytes)

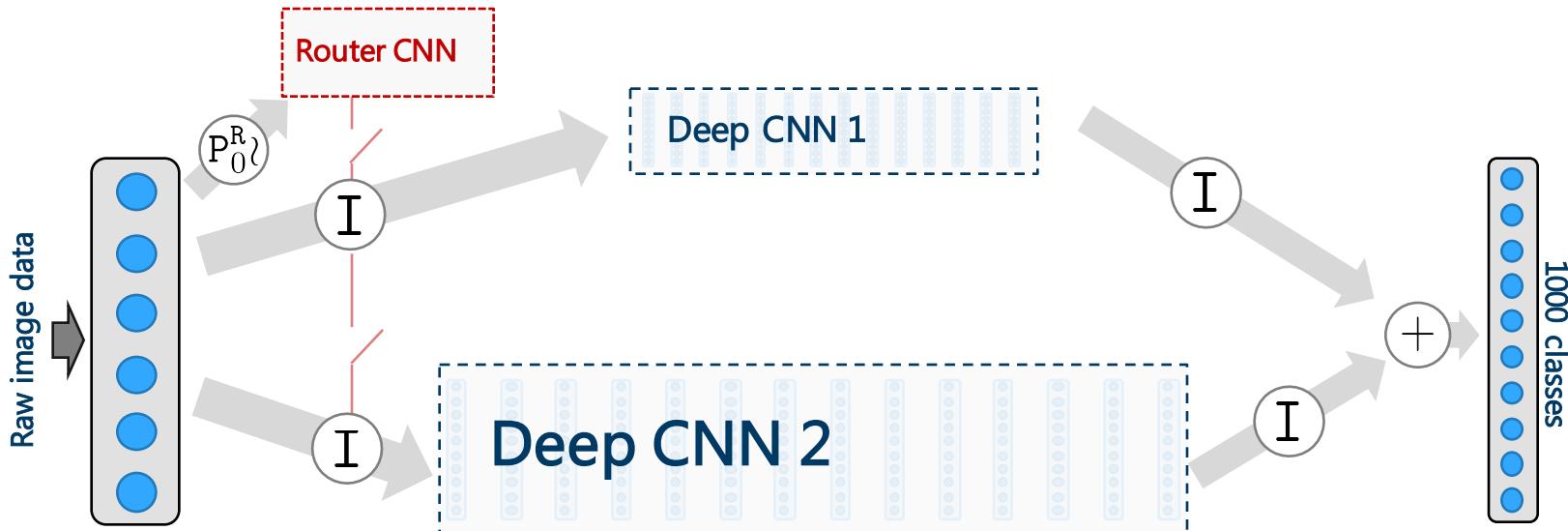
1/8 size, 1/7 cost, identical accuracy

Outline

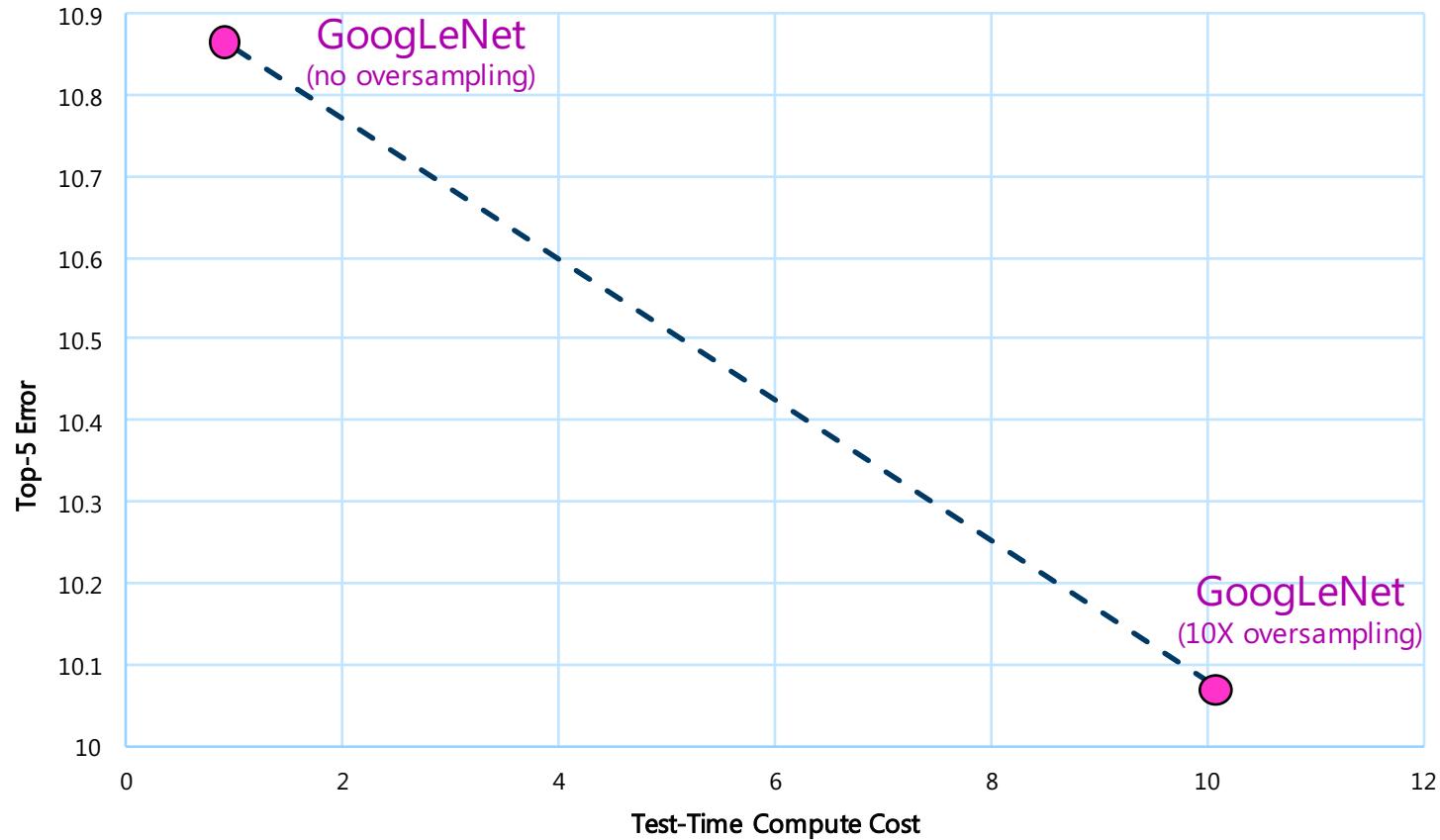
- **Introduction**
 - Trees, DAGs and deep neural networks (DNNs)
 - *Motivation:* Representing trees/DAGs as MLPs
 - *Motivation:* ReLUs and data routing
- **Conditional networks**
 - The model
 - Training via back-propagation
- **Experiments and results**
 - Conditional sparsification of a perceptron
 - Comparing architectures on ImageNet
 - Comparing architectures on CIFAR
 - **Conditional ensembles**
- **Conclusion**



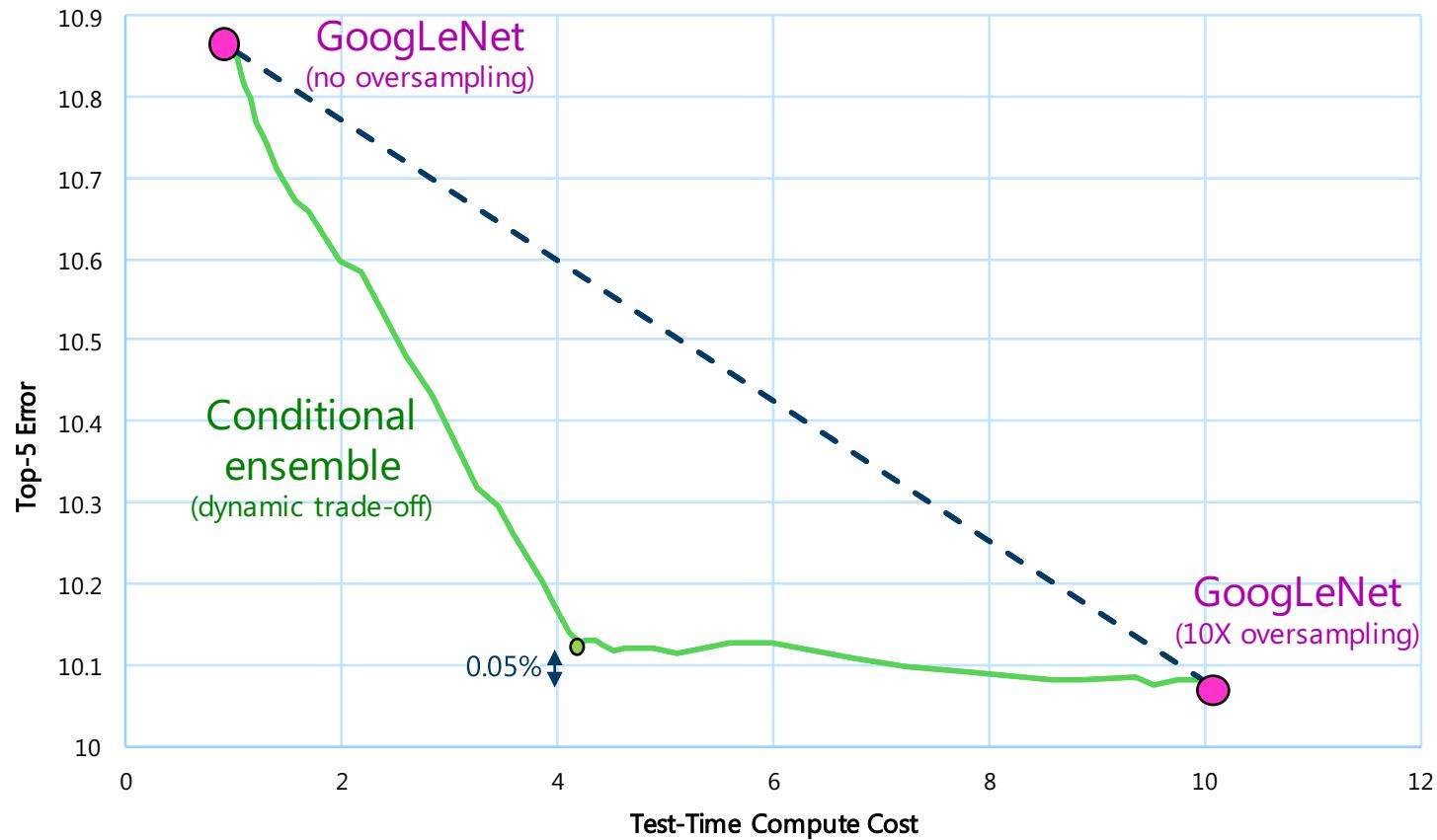
Conditional ensembles



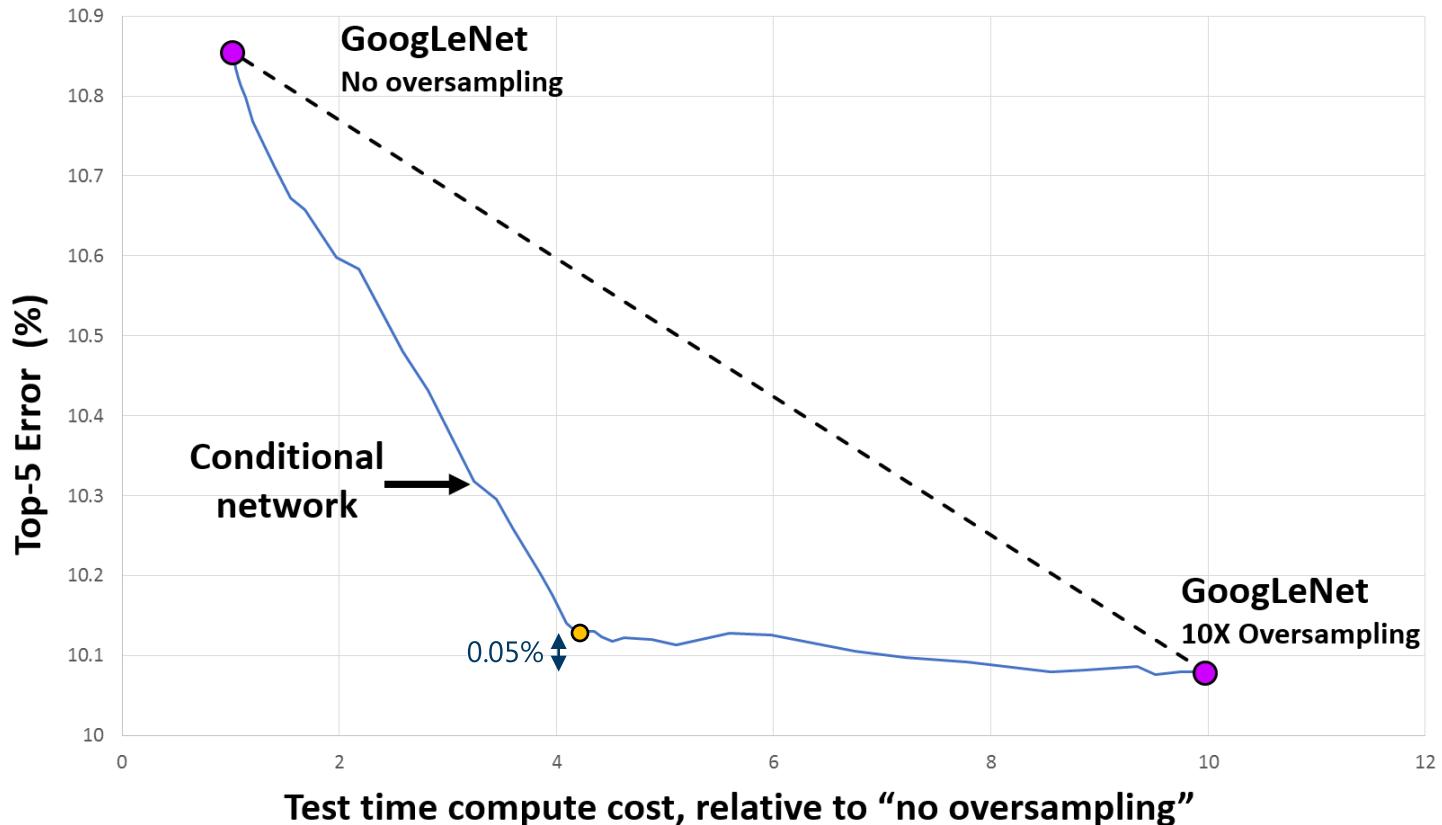
Conditional ensembles



Conditional ensembles



Conditional ensembles



Outline

- **Introduction**
 - Trees, DAGs and deep neural networks (DNNs)
 - *Motivation:* Representing trees/DAGs as MLPs
 - *Motivation:* ReLUs and data routing
- **Conditional networks**
 - The model
 - Training via back-propagation
- **Experiments and results**
 - Conditional sparsification of a perceptron
 - Comparing architectures on ImageNet
 - Comparing architectures on CIFAR
 - Conditional ensembles
- **Conclusion**

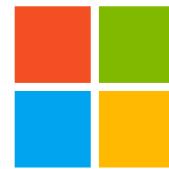


Conclusion

Conditional networks. A new, deep learning model with the following properties

- **Generalizes** decision forests, decision jungles and deep neural networks
- Can be **trained** entirely via back-propagation.
- **On-demand** selection of operating point = $f(\text{accuracy}, \text{efficiency}, \text{size})$
- Higher **interpretability** than deep neural networks



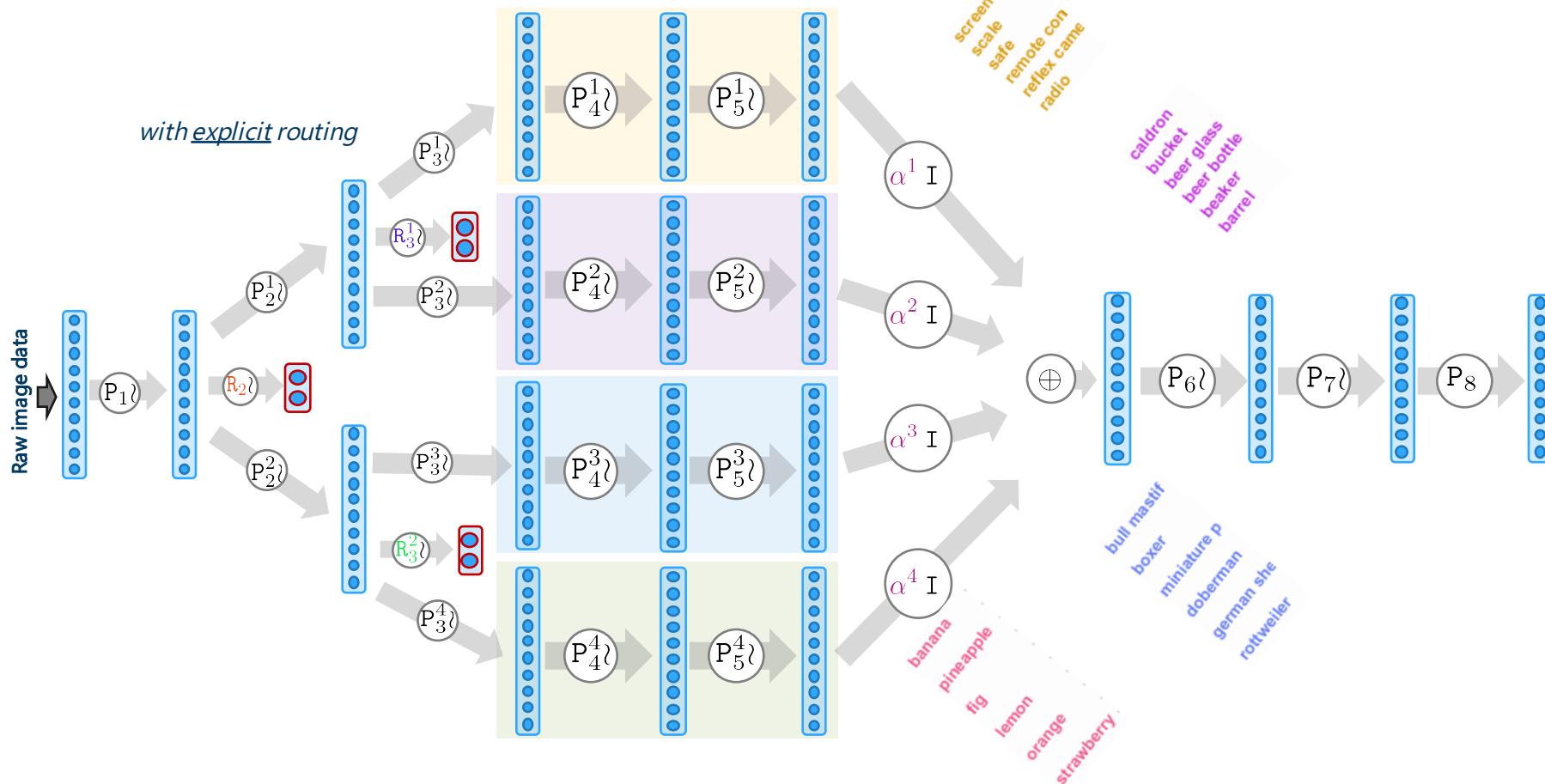


Microsoft

©2013 Microsoft Corporation. All rights reserved.

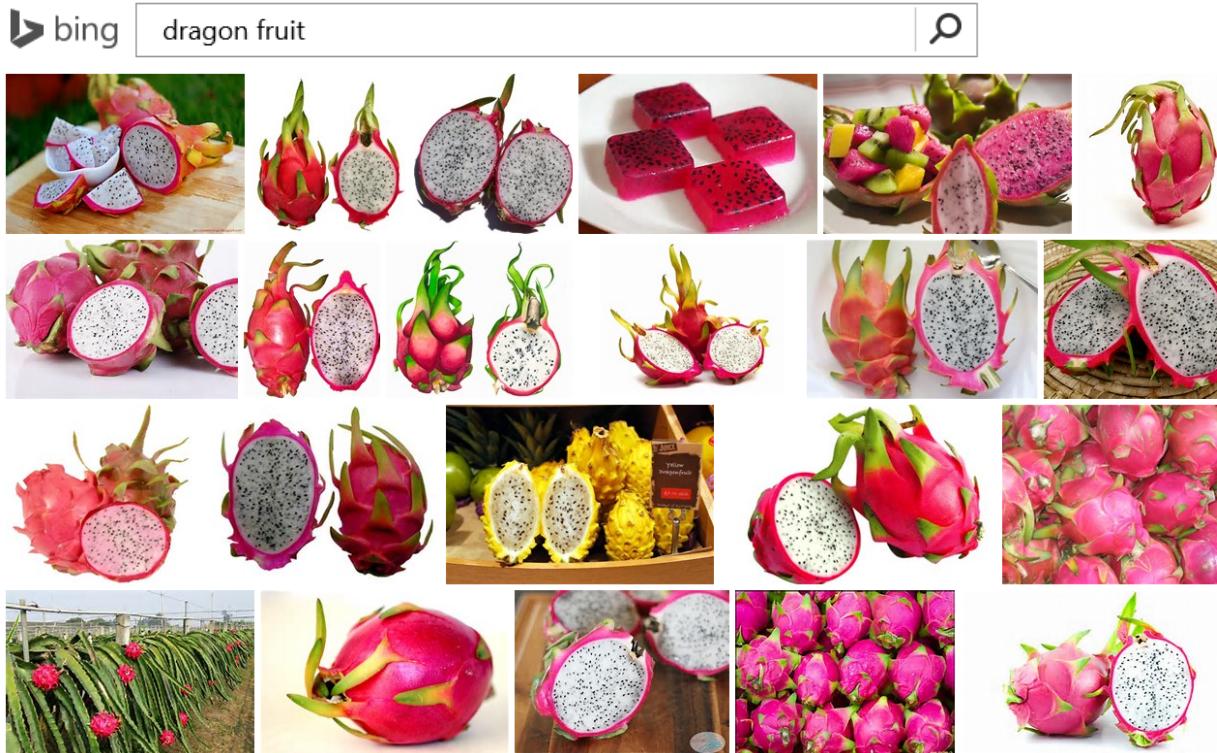
Appendix

On-demand learning of additional classes

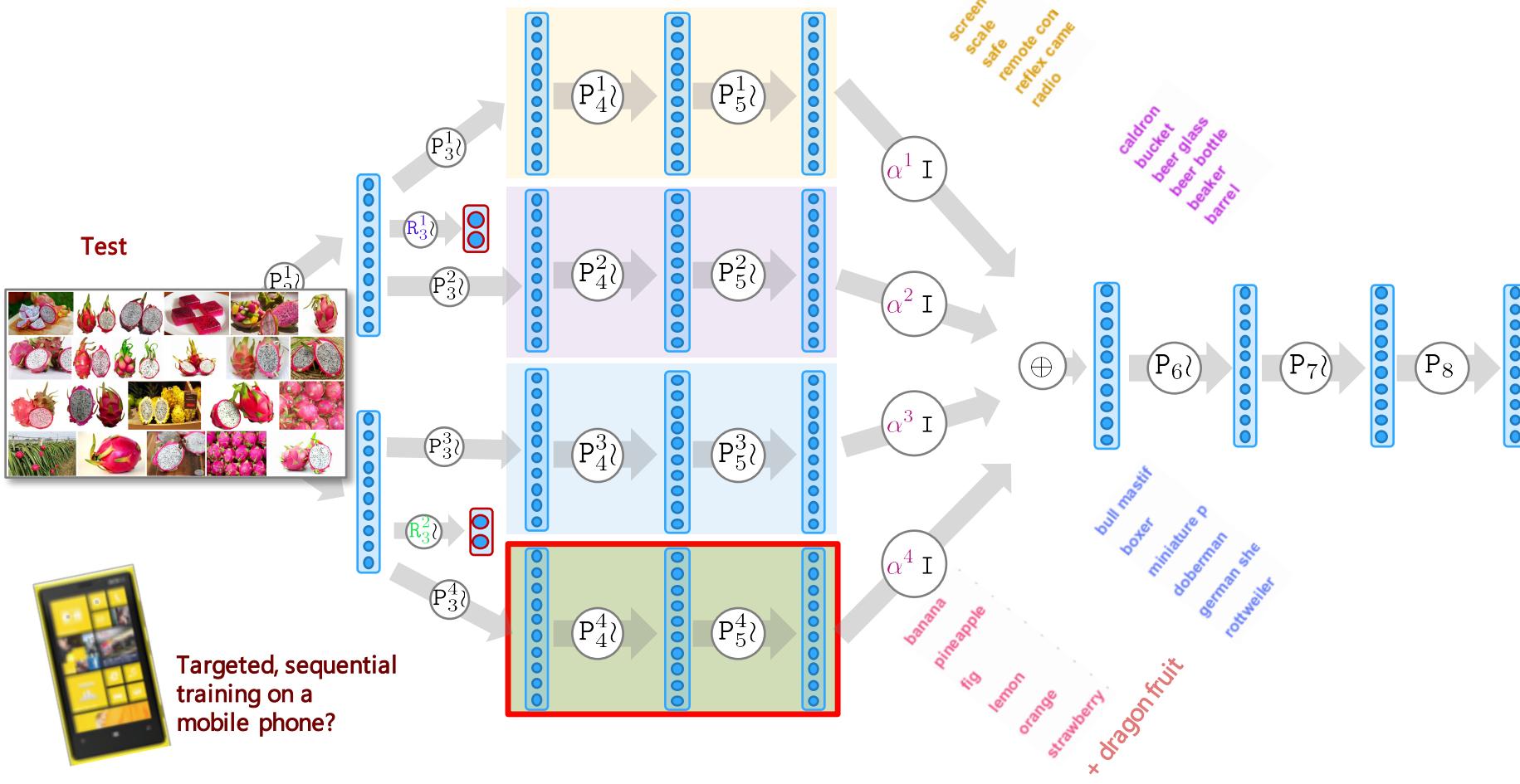


On-demand learning of additional classes

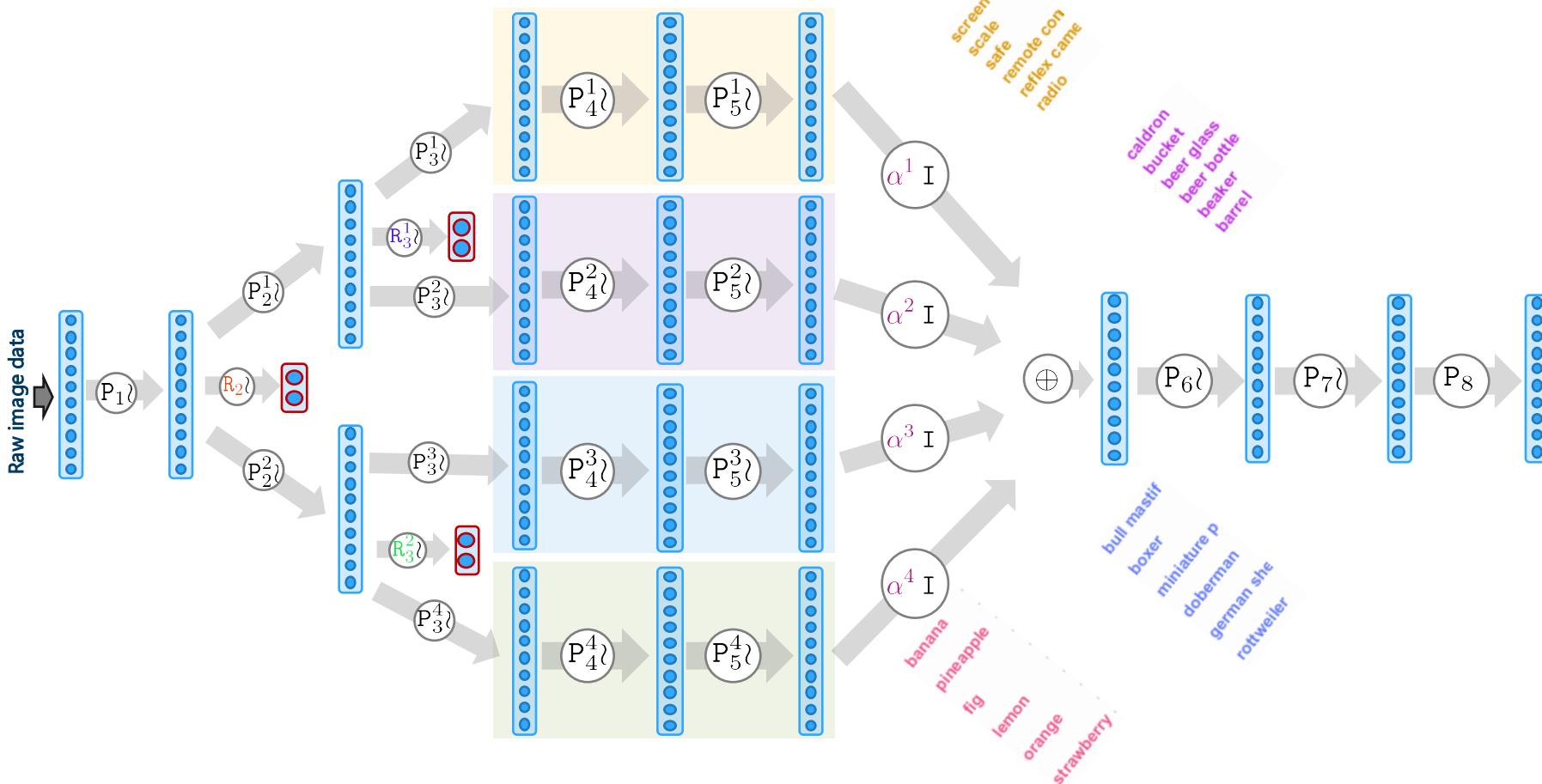
"Now I am also interested in dragon fruit"



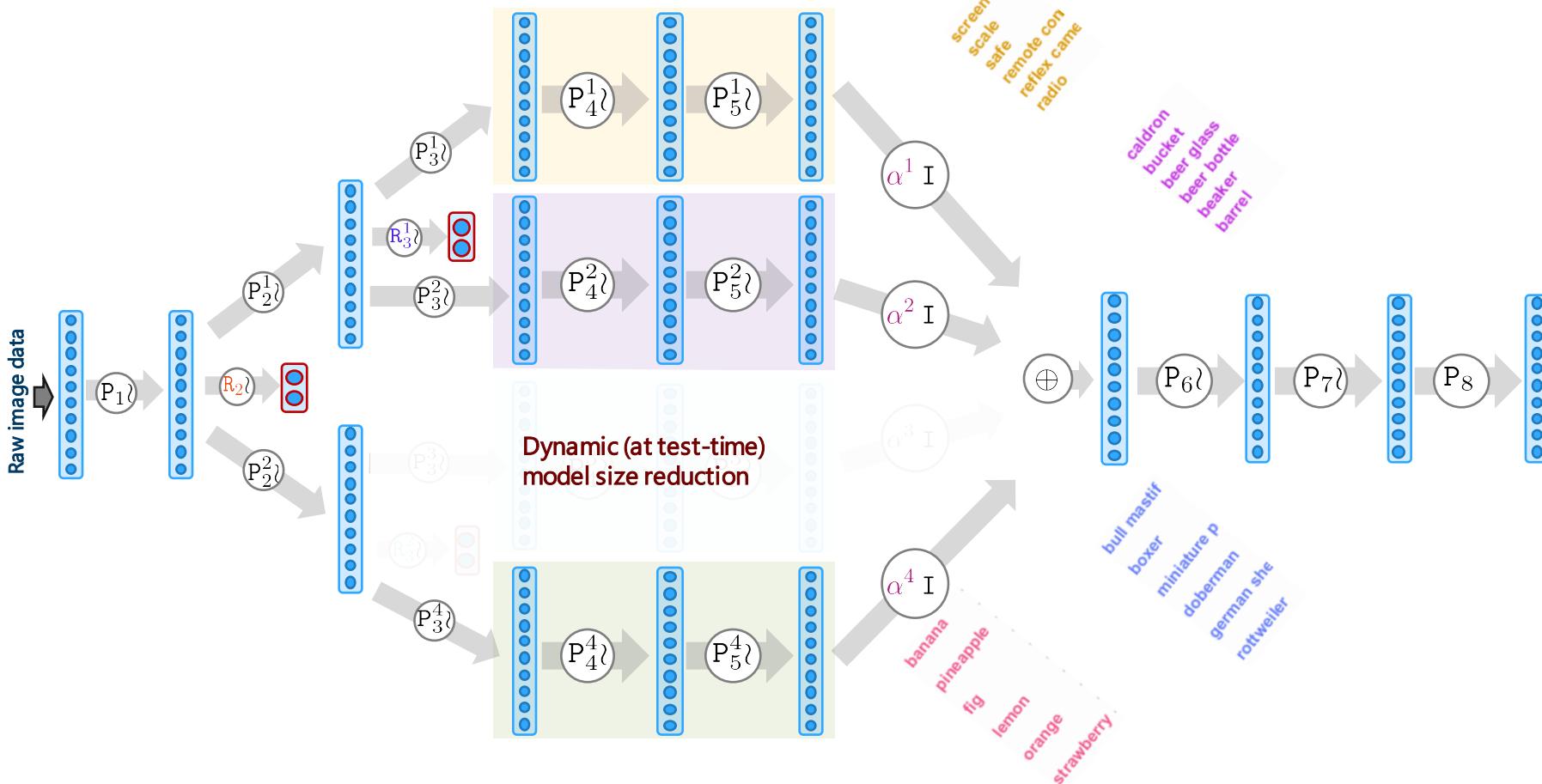
On-demand learning of additional classes



Dynamic structure adaptation



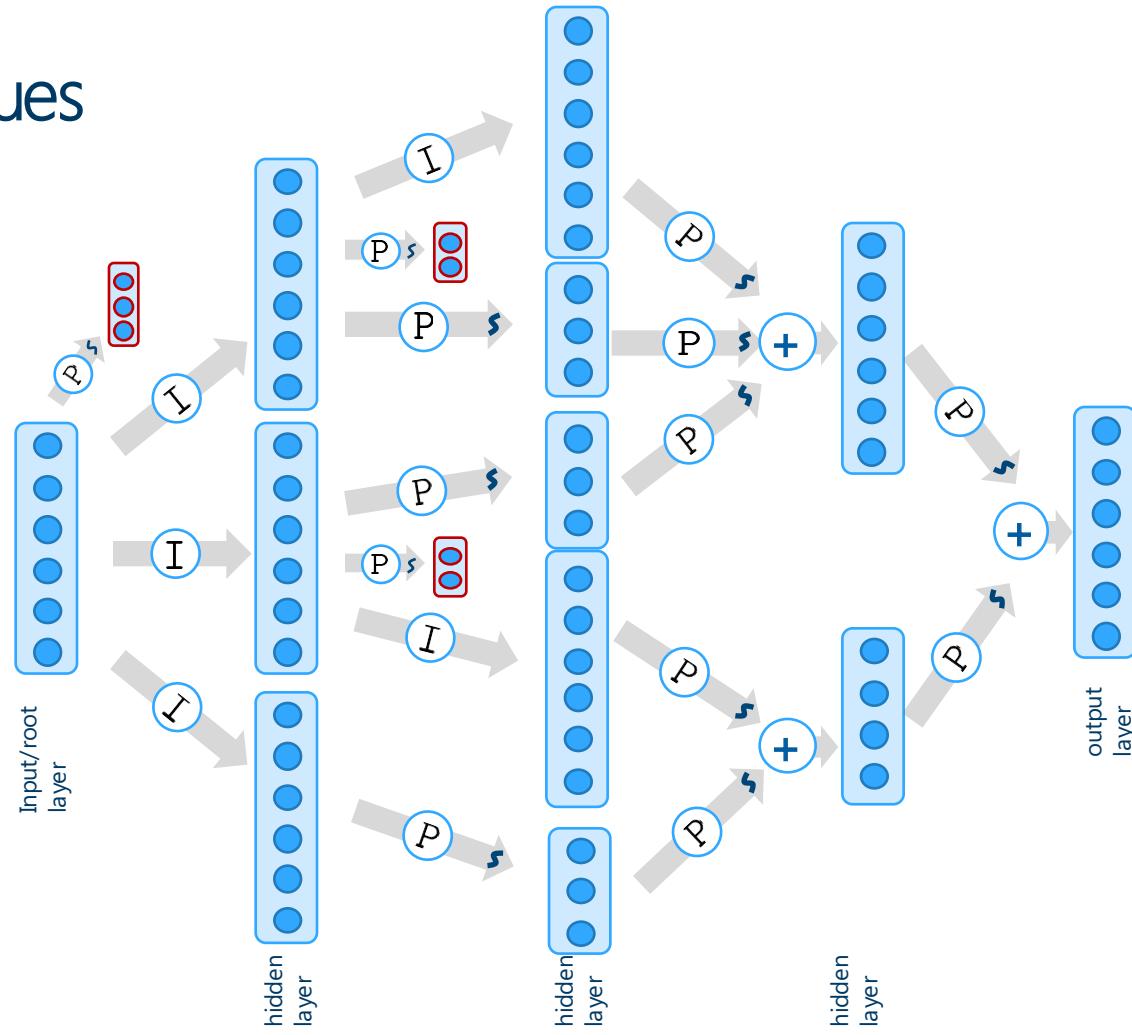
Dynamic structure adaptation



OLD UNUSED

Conditional networks some properties and issues

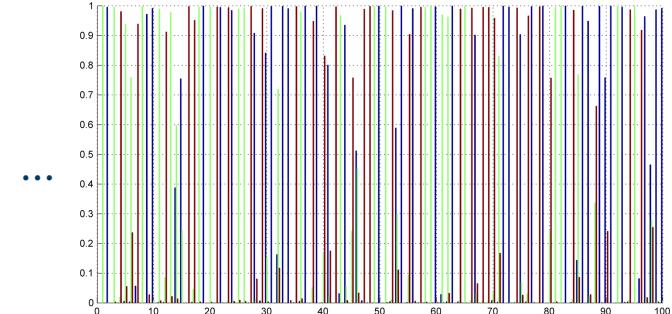
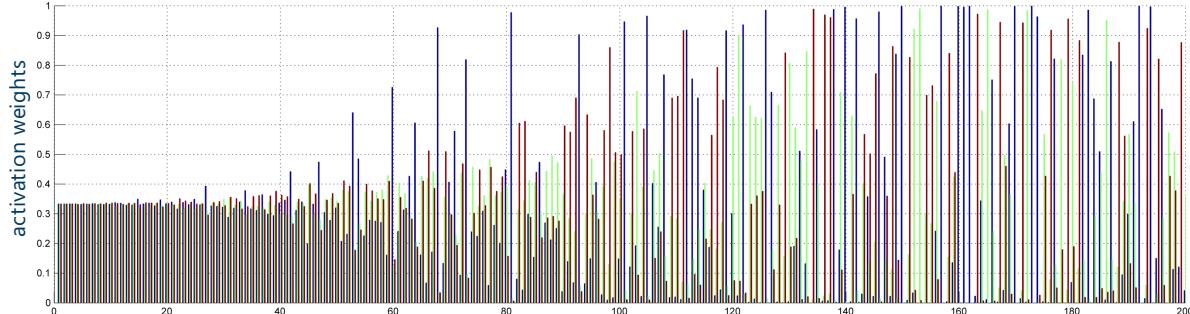
- If there is inherent structure in the input data we can exploit it and achieve saving. CuP can be seen as a way of sparsifying the weight (transformation) matrices
- Being able to learn the structure of the model is new compared to commonly used DNNs. Can we actually do it well?
- CuP vs MLP-forest.
 - In MLP forests we train a whole MLP at each node for data splitting. Here the MLP is distributed, a single layer at each node. It is trained directly for class ranking instead.
- Questions:
 - Can it all be learned via back-propagation? Perhaps the n-ary routing weights can be initialized greedily and then refined via gradient descent?
 - Can we / does it make sense to keep track of fractional class counts through nodes and empirical class distributions?
 - Is this way of achieving sparsification better than setting the lowest weights to 0 and use sparse matrix algorithms ?



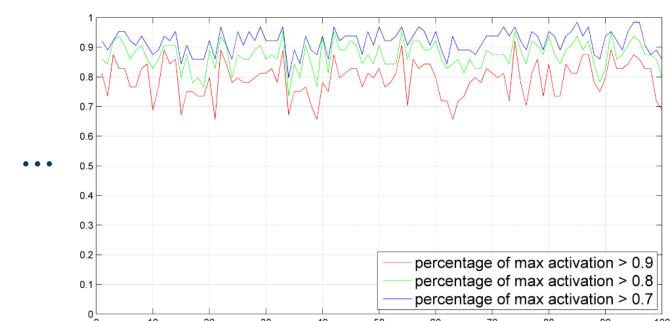
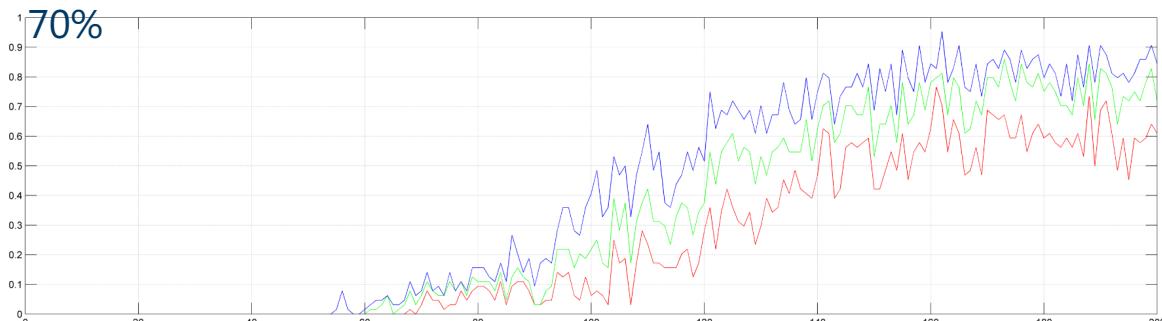
Training a classification-only CONNECT globally

From Darko

Examples of evolution of activation weights for the 3 different routes (R,G,B) during training



Influence of Individual Route: Percentage of maximum activation larger than thresholds of 90%, 80%,



Final percentage of max activation $> 0.90 = 0.7927$
Final percentage of max activation $> 0.80 = 0.8725$
Final percentage of max activation $> 0.70 = 0.9184$

Convolutional OpenVINO™ Super-classing Router

Training the router

Surrogate training energy

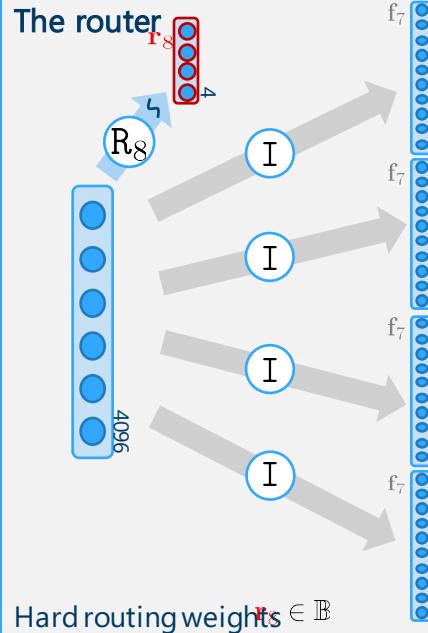
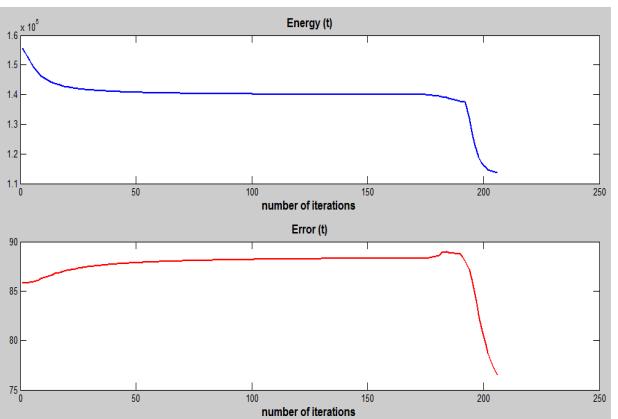
$$E = \sum_{i=1}^{>10^6} \sum_{j=1}^S w_j [s_i^* = s_j(\mathbf{v}_i)]$$

\mathbf{v}_i is the 4096-dimensional descriptor for image
 s_j denotes a super-class, out of S of them (8)

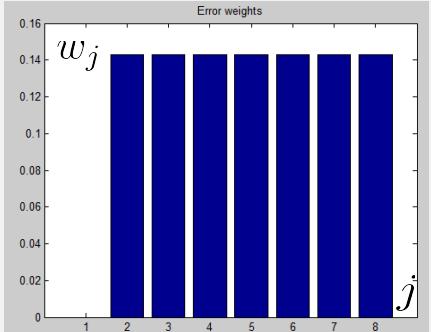
$s_j(i) \text{ s.t. } a_{j-1} > a_j > a_{j+1}$

a_j denotes activation for superclass j

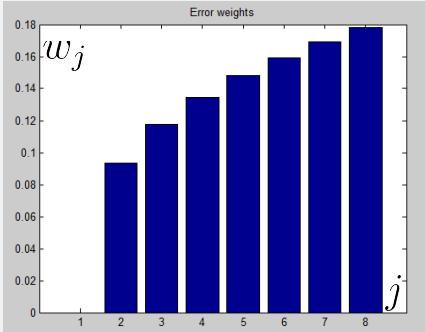
Key to have randomized, numerical gradient descent work is to have a smooth surrogate training energy.



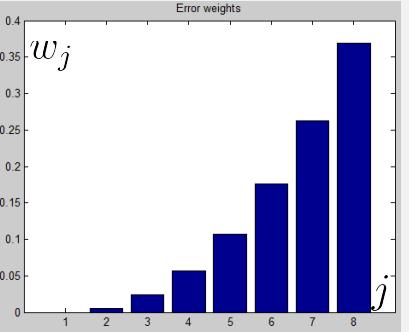
Conventional super-class classification error (top-1)



Smoother energy



Smoother energy encouraging mass into top-ranked branches

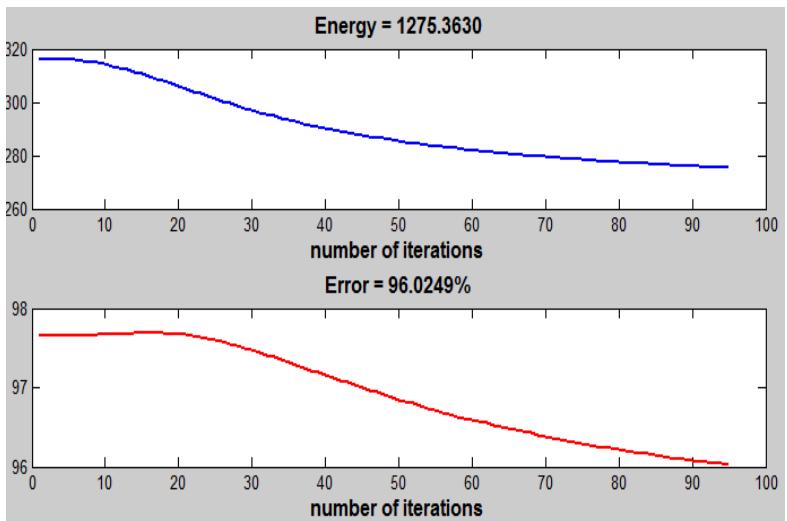


Slower convergence but lower local minimum?

Training the router

Iterating

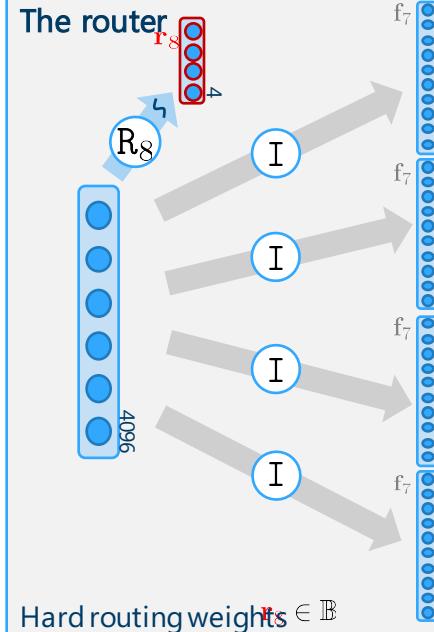
- PSO, Particle swarm optimization (on a smoother surrogate energy), and
- Class to superclass reassignment



The K-Hyperplanes optimization algorithm:

1. Initialize class-to-superclass assignments, e.g. via Kmeans on fc7. (The Kmeans centres are not used, just the class assignments)
2. Initialize router projection with random values
3. numerical gradient descent on the router matrix to minimize soft energy for ranking branches (in the spirit of random forests).
4. reassign classes to superclasses using current router projection and go to 3.

The above works well, but initializing the router projection (point 1.) from the already available P8 matrix leads to faster convergence.



Conditional Networks: summary

Contributions

- Demonstrated equivalence of trees, dags and two-layer perceptrons.
- Conditional Networks generalize neural networks and n-ary decision forests and jungles.
- With the right choices of model parameters we can potentially obtain saving in terms of compute and memory.
- We should be able to train the whole model globally via back-propagation.
- We now have a means of regularizing MLP training so as to prefer a particular kind of efficient structure, via lateral inhibition.
- We should be able to also learn (part of) the network structure rather than defining it all by hand.
- Combining this with the use of separable filters promises further gains
- Easily implementable within existing GPU-based codebases such as Caffe.

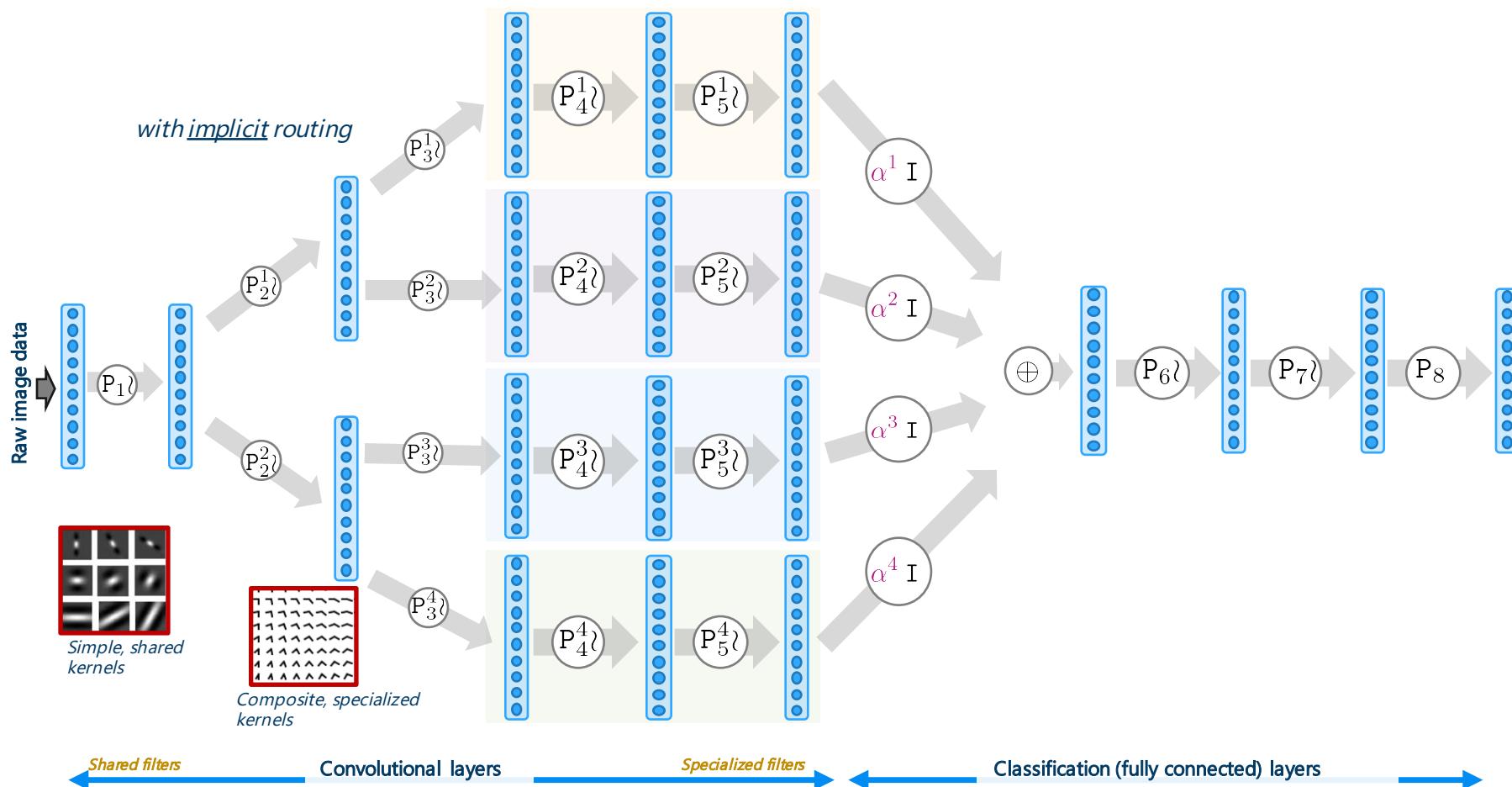
Rewrite this

Summary 2

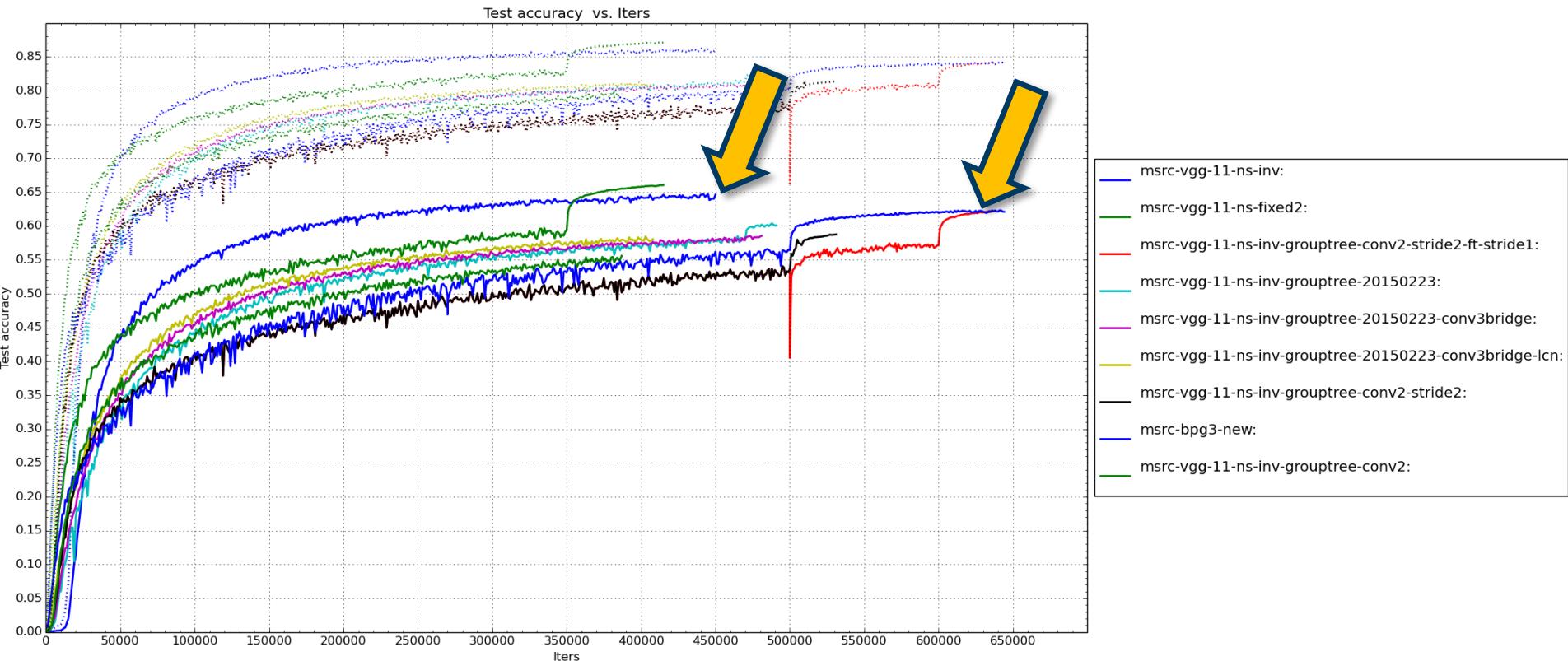
Rewrite this

- Decision trees/DAGS can be represented as a special form of 2-layer perceptrons and thus they can be trained via back-propagation.
- Deep Networks (with ReLu activations) implement a special routing mechanism. Such routing can be made explicit in the model architecture to achieve higher efficiency. This gives rise to a new learning model that we call "Conditional Networks".
- In summary, Conditional Networks represent a new model which bridges the gap between decision trees/DAGs and (deep) neural networks. They can represent both. Some model choices allow us to get the best in terms of efficiency-vs-accuracy.
-

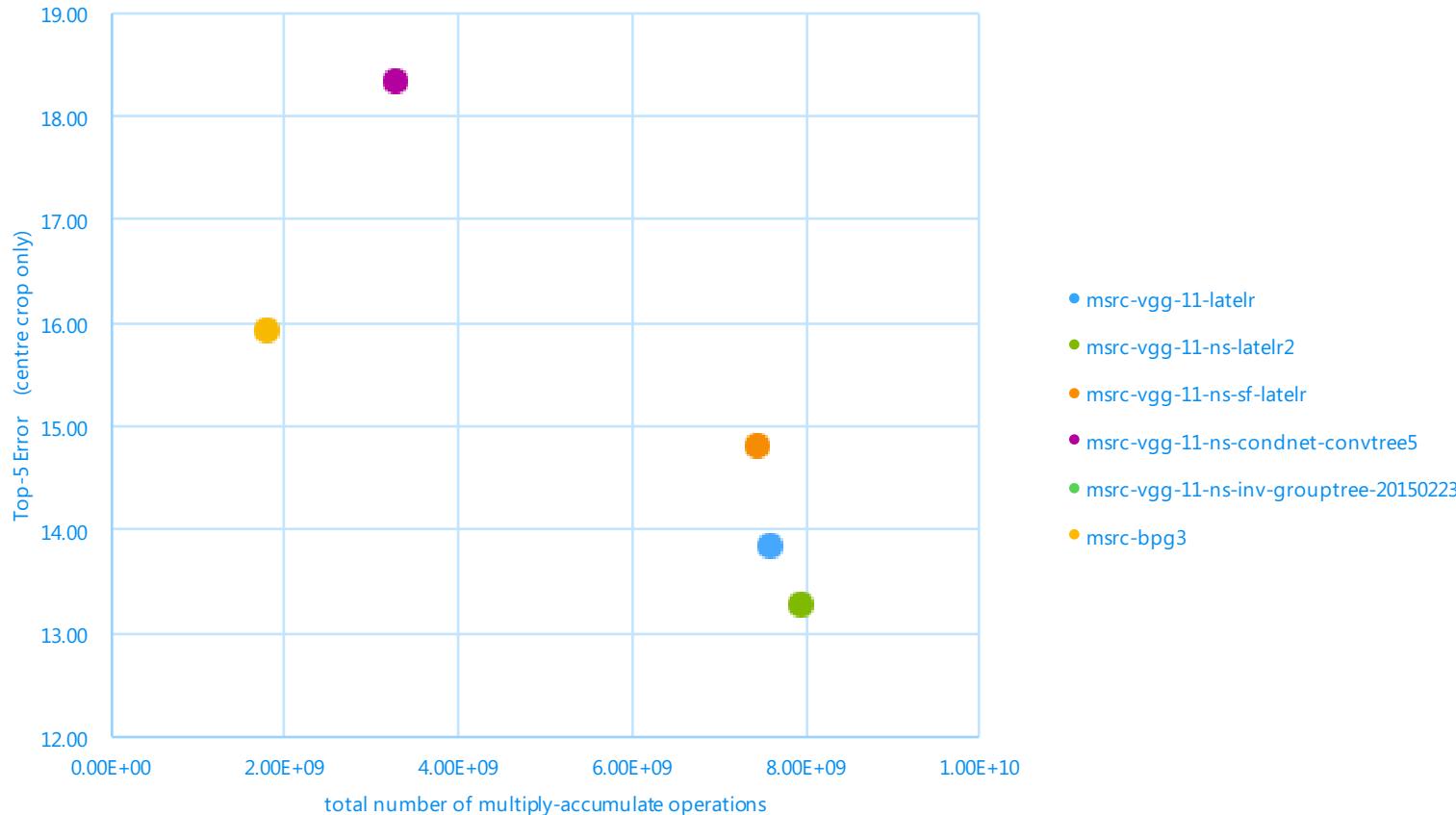
Results on ImageNet (1,000 classes)



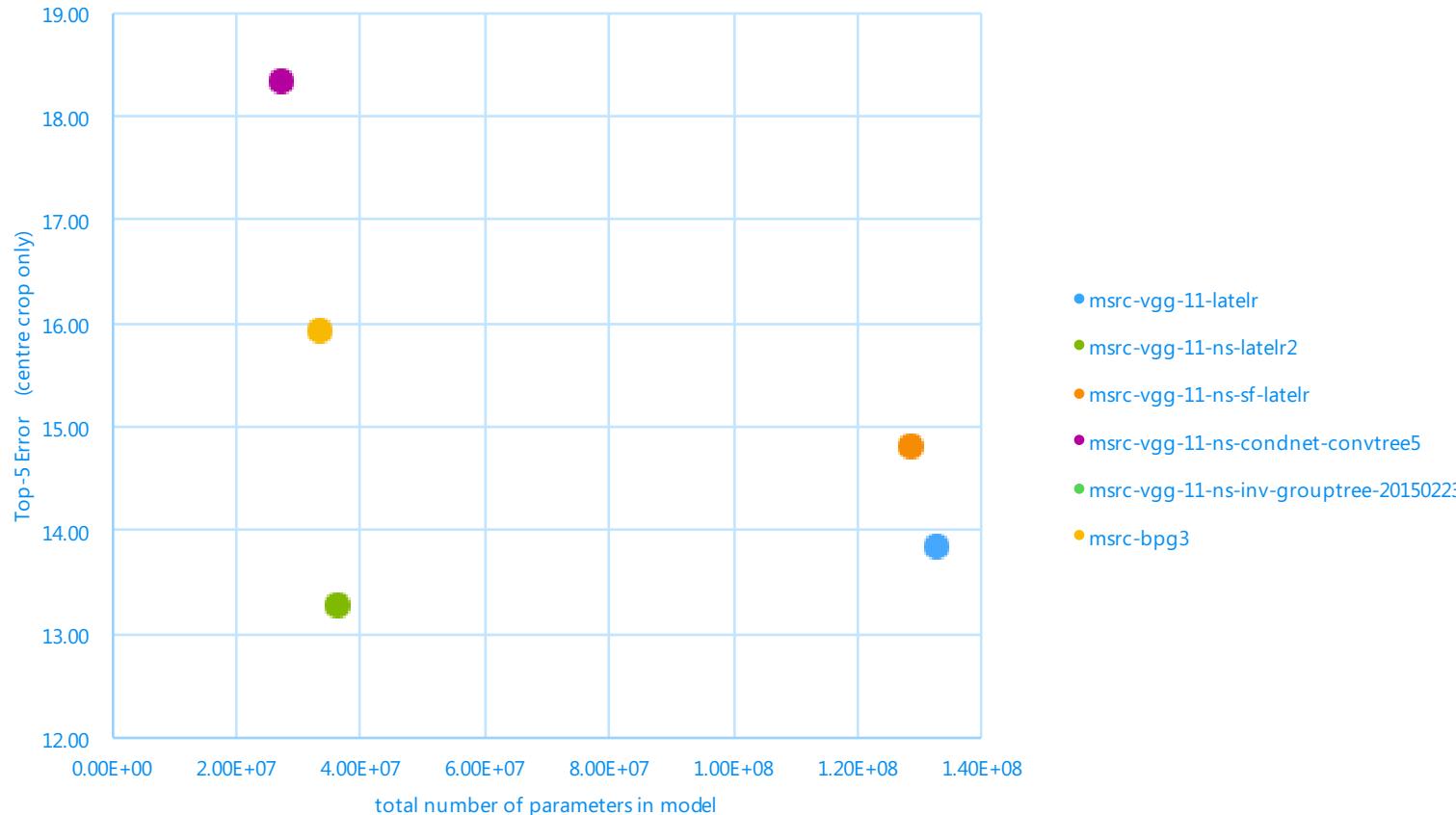
Results on ImageNet (1,000 classes)



Results on ImageNet (1,000 classes)



Results on ImageNet (1,000 classes)



Main results

- **Claim 1. Greater efficiency, on the fly**

- In a DNN the point (accuracy,efficiency) is fixed and unchangeable unless we train a new DNN.
- In a CONNET, a whole curve is available for a single trained CONNET. The optimal (accuracy,efficiency) can be changed on the fly, at TEST time, based on the dynamically changing application requirements.

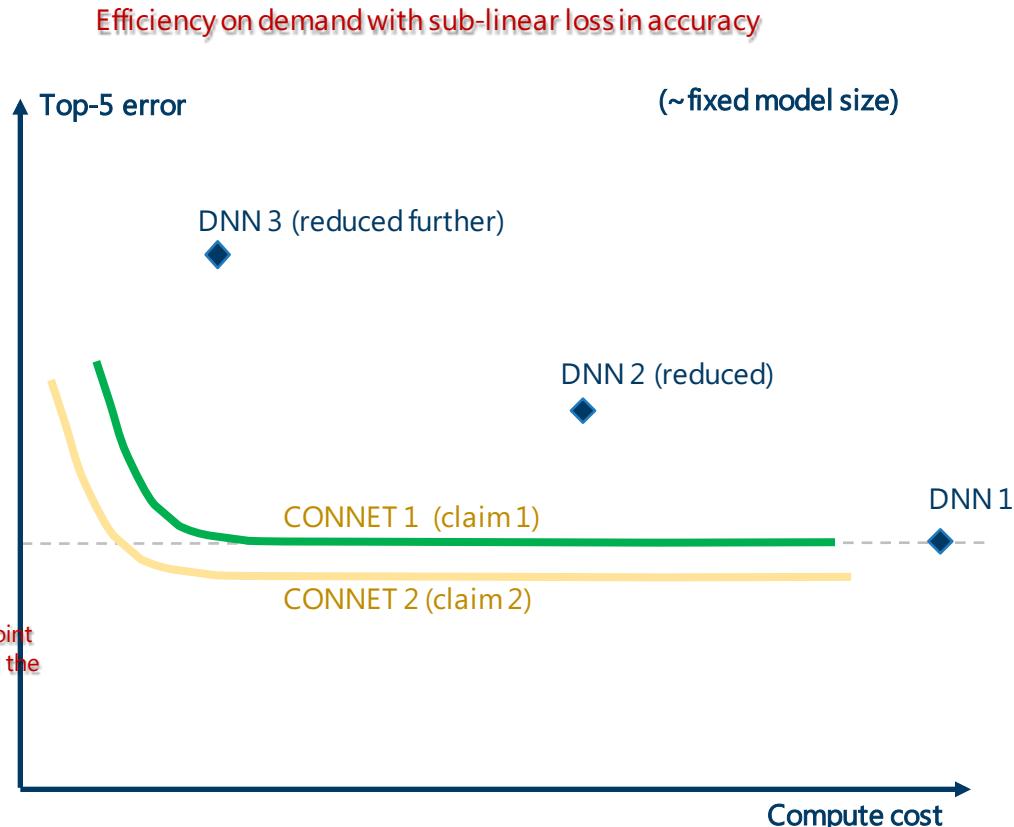
- **Claim 2. Greater efficiency and accuracy**

- training a CONNECT from scratch also achieves **better generalization?**

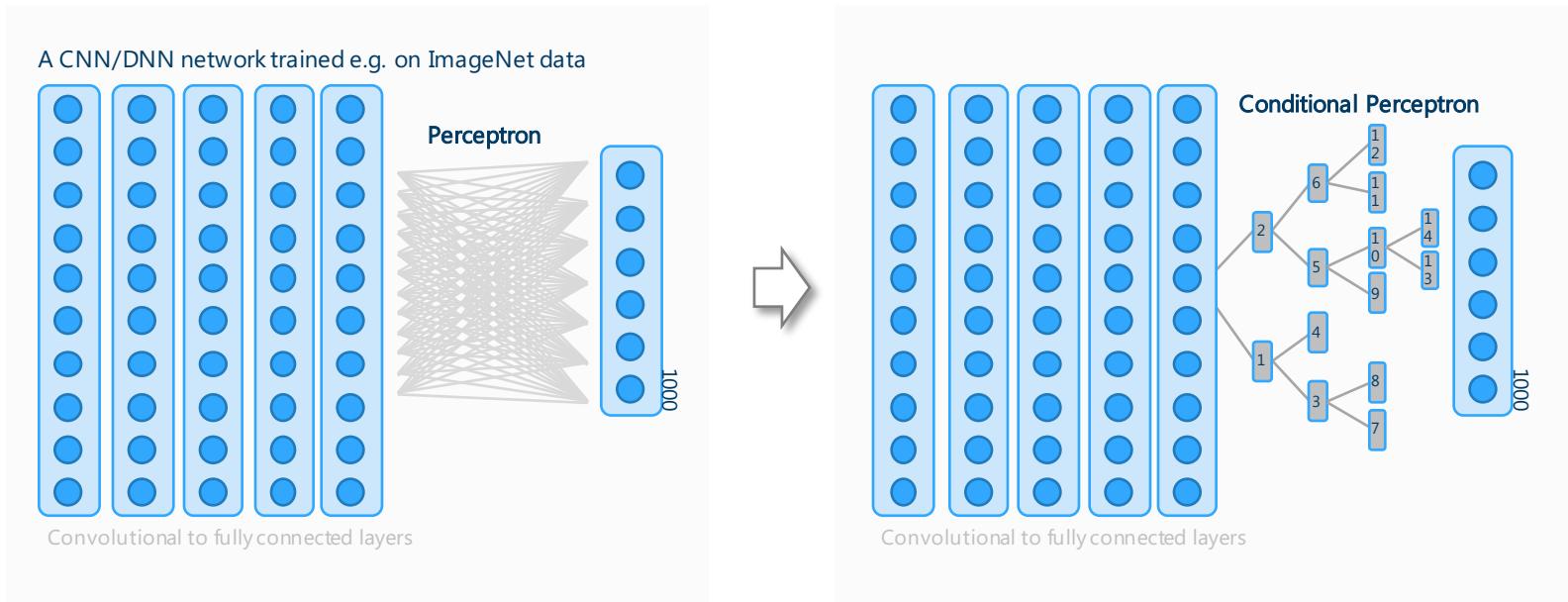
Efficiency on demand with sub-linear loss in accuracy

example in a phone app, as day progresses the battery is drained. At some point we automatically adjust the operating point so as to reduce the accuracy and the test time cost, thus reducing the speed of battery consumption, on the fly.

We need to make sure we speed up also the early convolutional layers!



Conditional sparsification of a perceptron

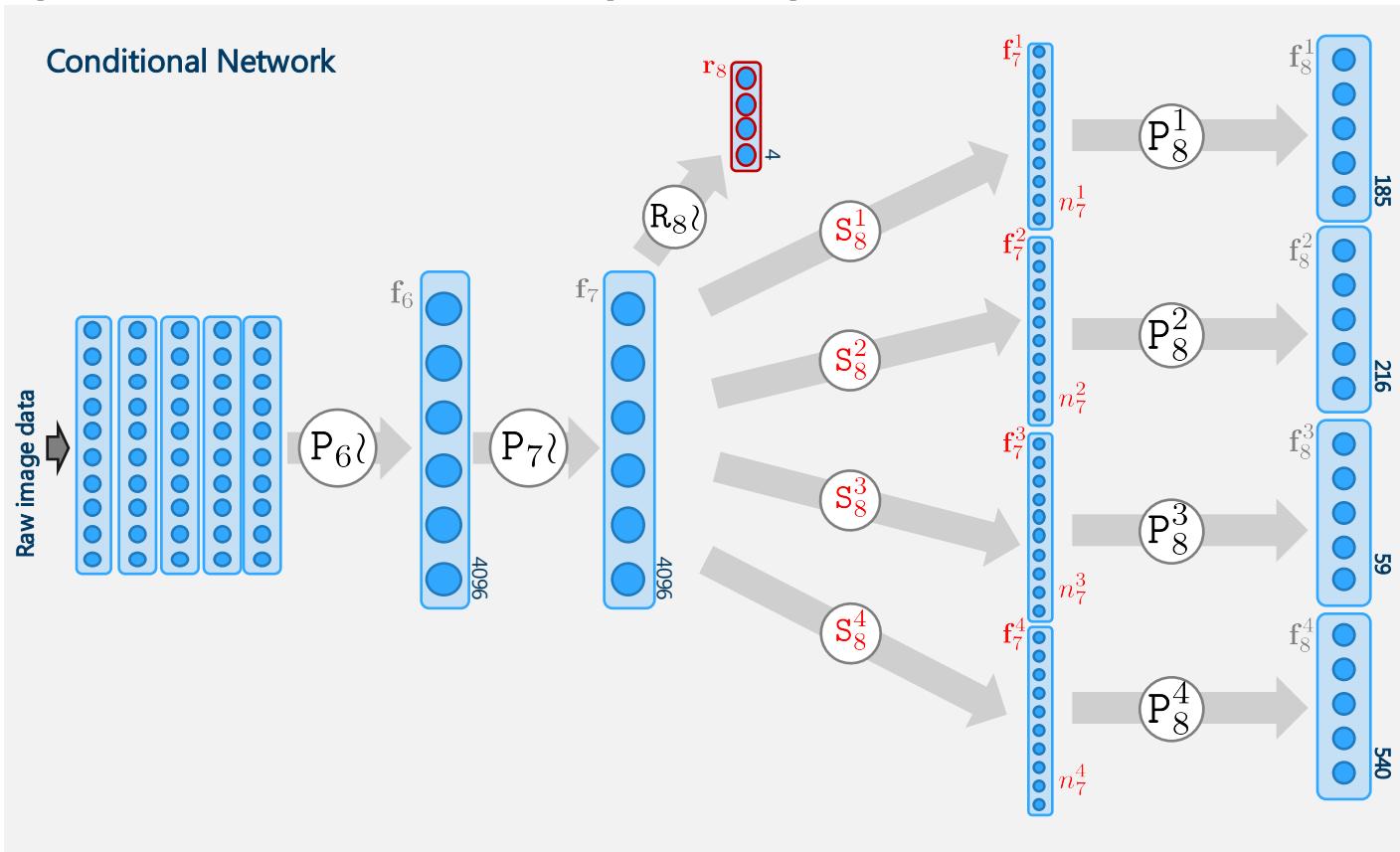


Conditional sparsification of a perceptron

Results 2. model compression

Experiment demonstrate effective sparsification of a single perceptron.
No deep architecture yet.

Now reducing model complexity by removing unnecessary features from the various tree branches



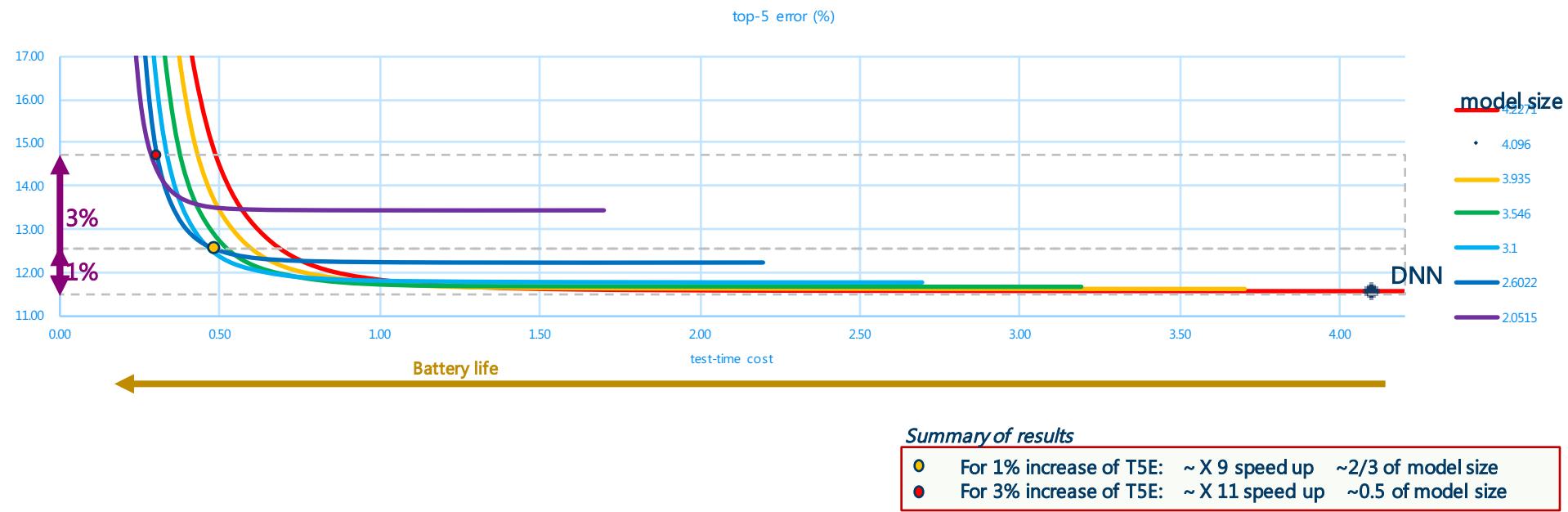
$$\text{Hard routing weights } r_8 \in \mathbb{B}$$

$$\cup f_7^j = f_7 \quad n_7^j \leq 4096 \quad \sum_j n_7^j \geq 4096 \quad \sum_j n_8^j = 1000$$

Conditional sparsification of a perceptron

Results 2. model compression (final layer only)

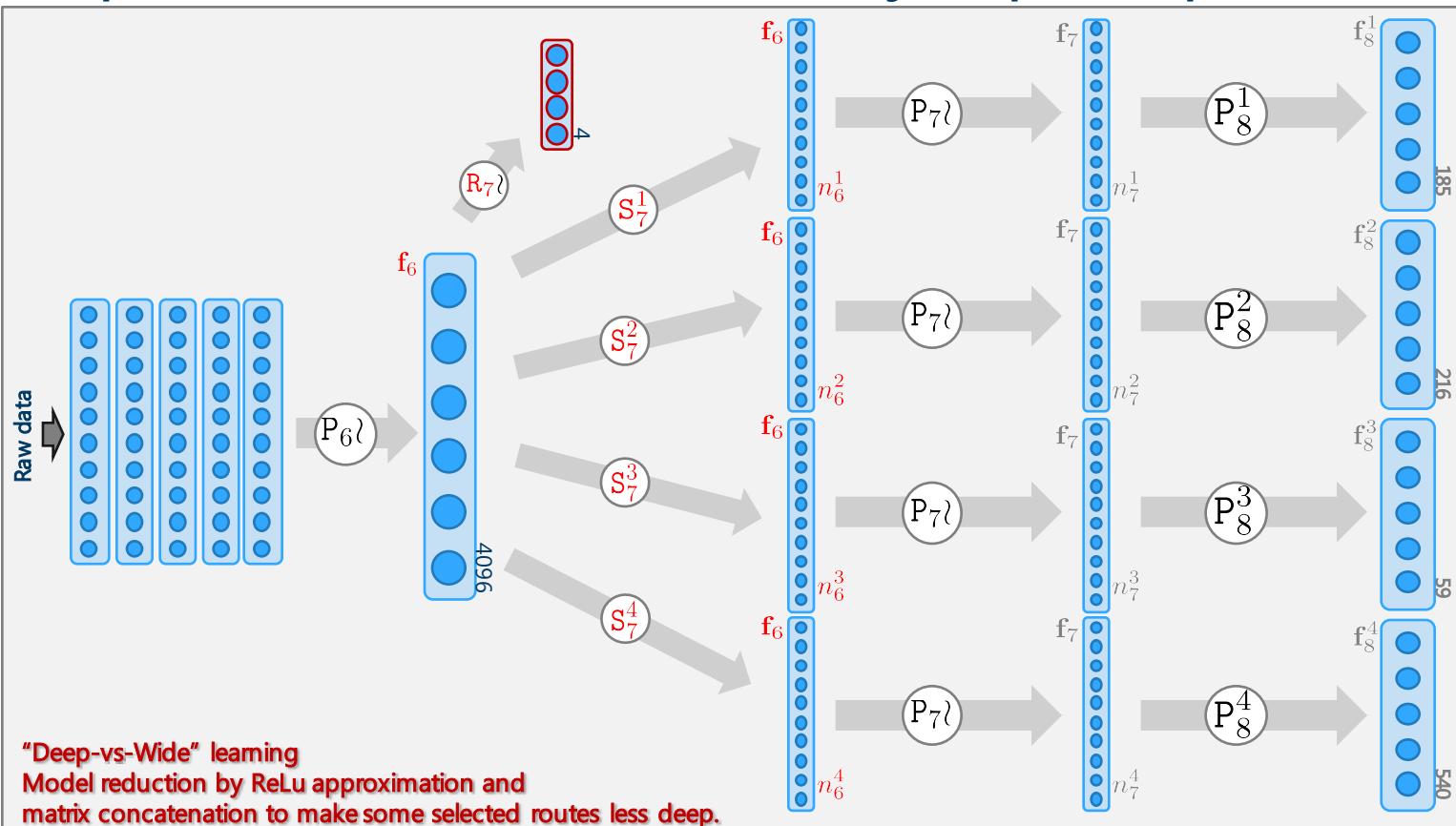
Experiment demonstrate effective sparsification of a single perceptron. No deep architecture yet.



Conditional sparsification of a multi-layer perceptron

Unzipping the net

$$\begin{aligned}
 f_8^i &= f_7 P_8 \\
 f_7 &= \sigma(f_6 P_7) \\
 f_8^i &= \sigma(f_6 P_7) P_8^i \\
 b_7^i &= [(f_6 P_7) > 0] \in \mathbb{B}^{4K} \\
 \tilde{P}_t^i &= P_7 \text{ diag}(b_7^i) \\
 f_7^i &:= f_6 \tilde{P}_7^i \approx f_7 \\
 f_8^i &= f_6 \tilde{P}_7^i P_8^i \\
 &\quad (\text{transposed})
 \end{aligned}$$

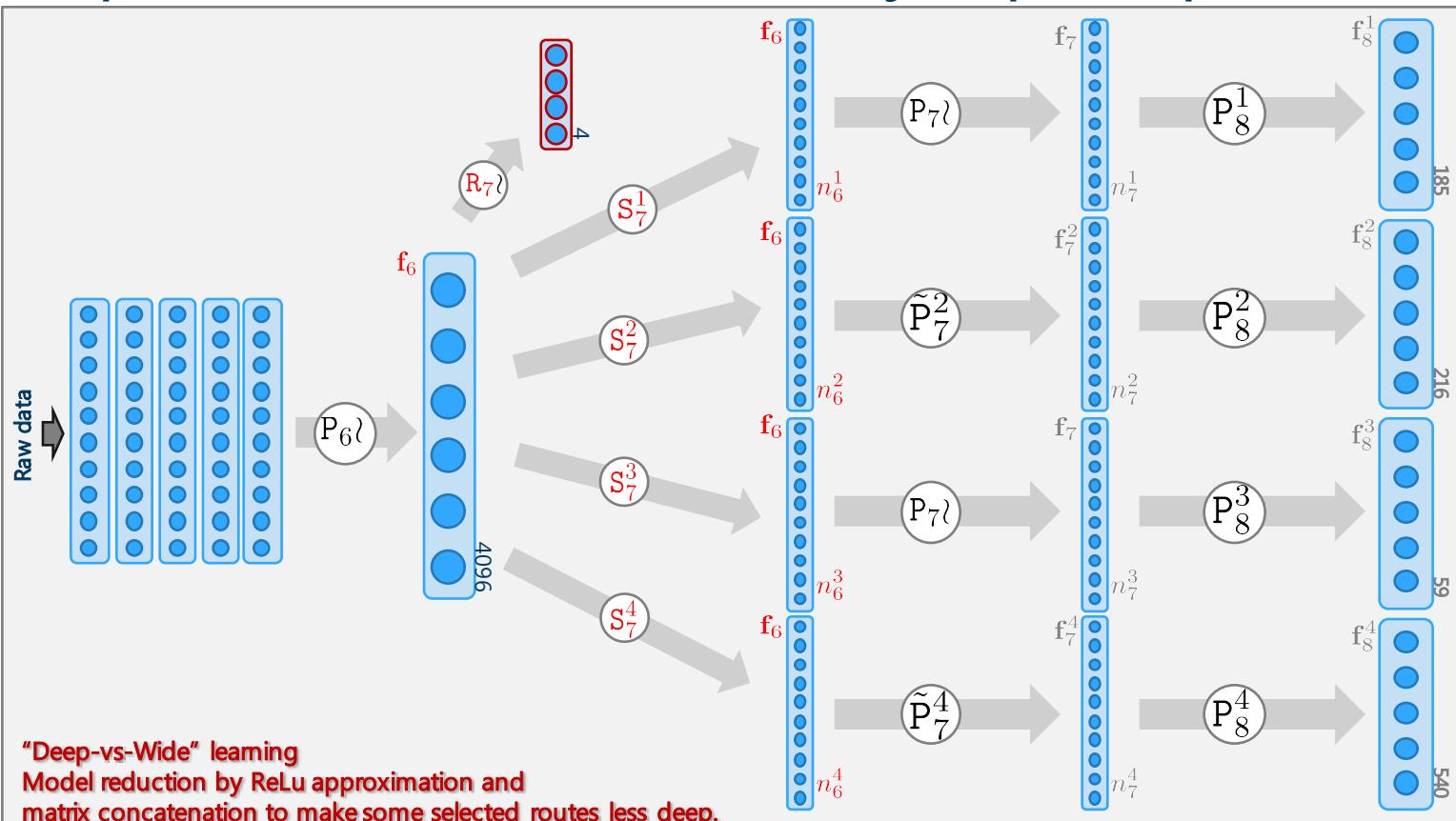


$$\bigcup_j f_7^j = f_7 \quad \bigcap_j f_7^j = \emptyset \quad n_7^j \leq 4096 \quad \sum_j n_7^j \geq 4096 \quad \sum_j n_8^j = 1000$$

Conditional sparsification of a multi-layer perceptron

Unzipping the net

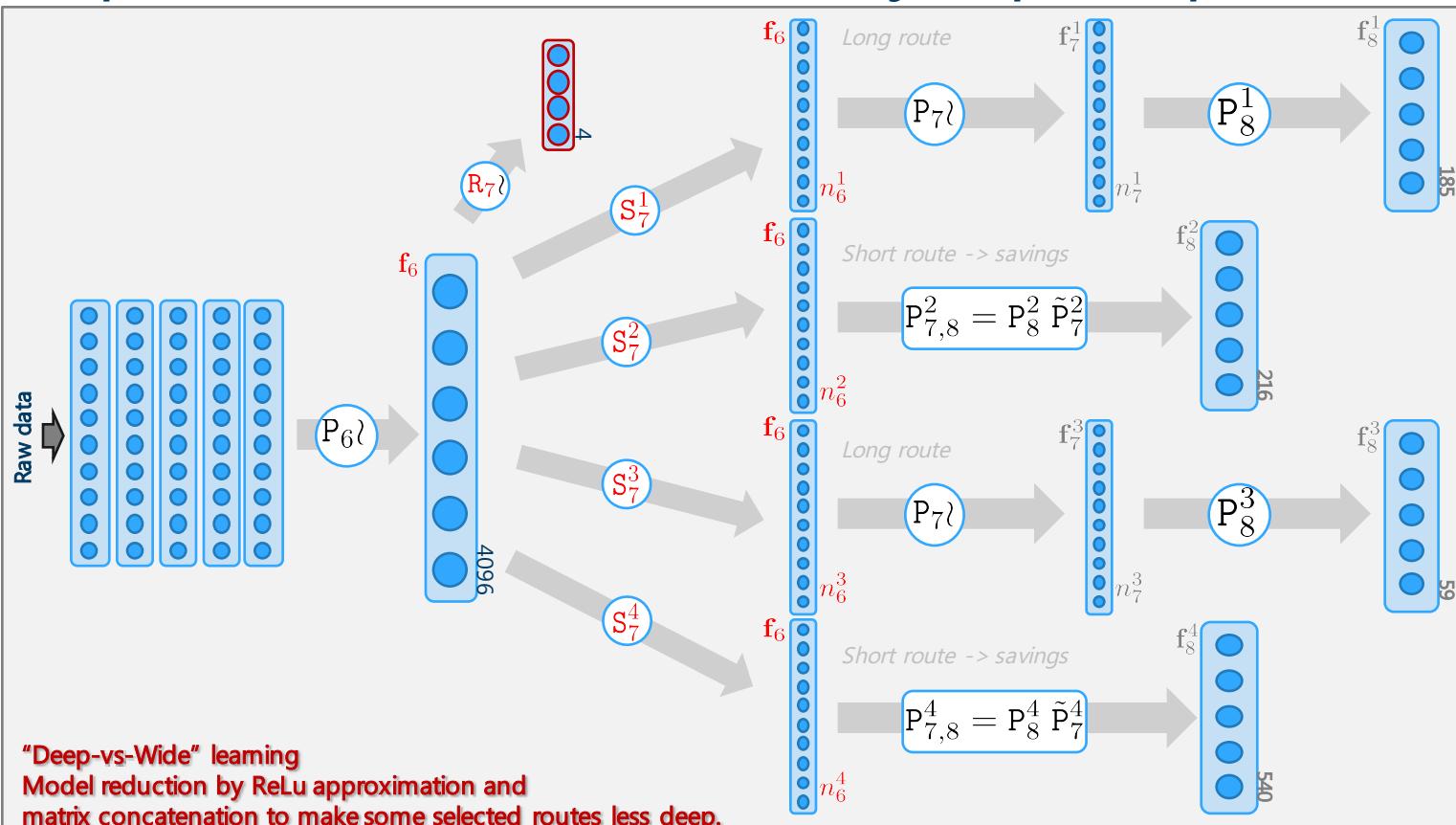
$$\begin{aligned}
 f_8^i &= f_7 P_8 \\
 f_7 &= \sigma(f_6 P_7) \\
 f_8^i &= \sigma(f_6 P_7) P_8^i \\
 b_7^i &= [(f_6 P_7) > 0] \in \mathbb{B}^{4K} \\
 \tilde{P}_t^i &= P_7 \text{ diag}(b_7^i) \\
 f_7^i &:= f_6 \tilde{P}_7^i \approx f_7 \\
 f_8^i &= f_6 \tilde{P}_7^i P_8^i \\
 &\quad (\text{transposed}) \\
 \tilde{P}_7^i P_8^i &\in \mathbb{R}^{4K \times N_i}
 \end{aligned}$$



Conditional sparsification of a multi-layer perceptron

Unzipping the net

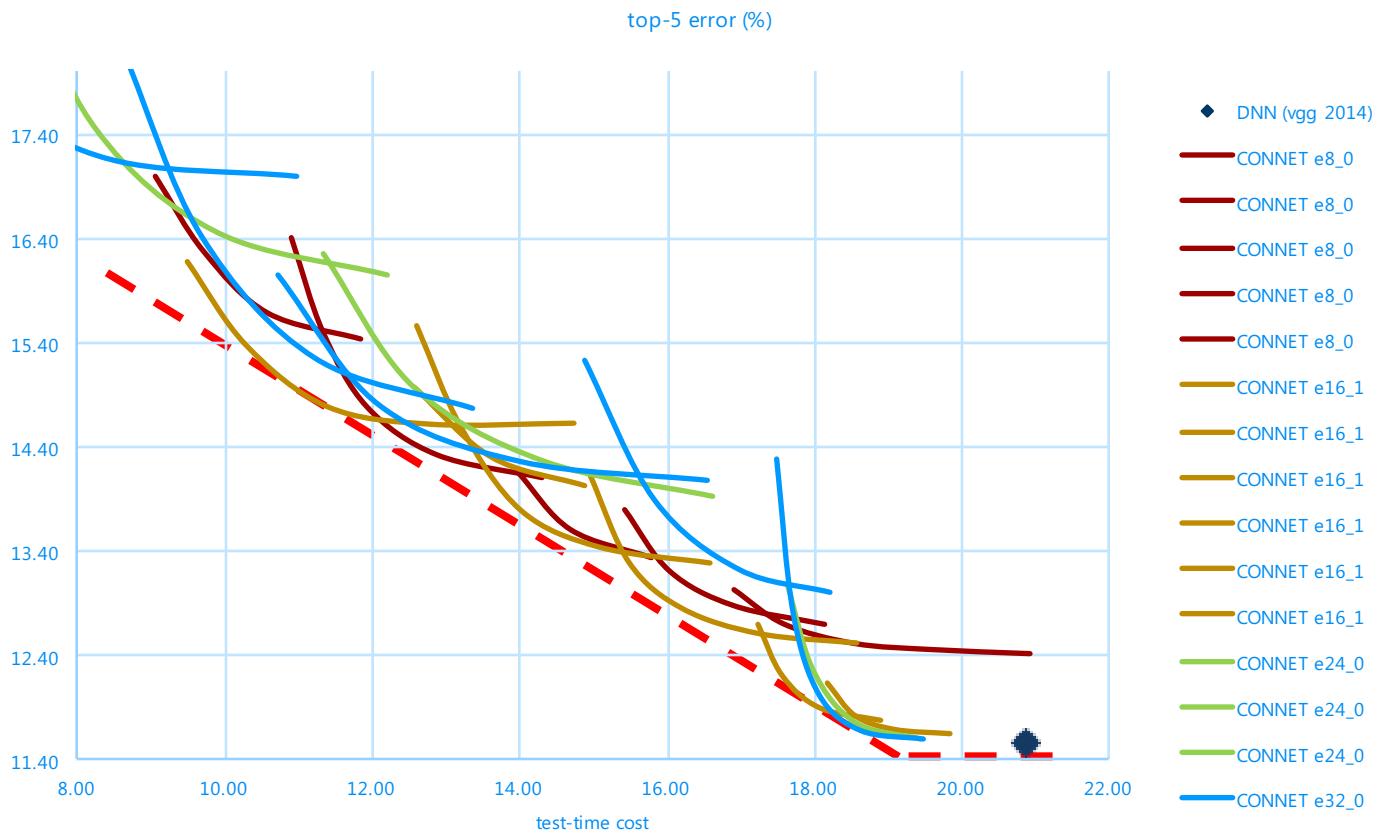
$$\begin{aligned}
 f_8^i &= f_7 P_8 \\
 f_7 &= \sigma(f_6 P_7) \\
 f_8^i &= \sigma(f_6 P_7) P_8^i \\
 b_7^i &= [(f_6 P_7) > 0] \in \mathbb{B}^{4K} \\
 \tilde{P}_t^i &= P_7 \text{ diag}(b_7^i) \\
 f_7^i &:= f_6 \tilde{P}_7^i \approx f_7 \\
 f_8^i &= f_6 \tilde{P}_7^i P_8^i \\
 &\quad (\text{transposed}) \\
 \tilde{P}_7^i P_8^i &\in \mathbb{R}^{4K \times N_i}
 \end{aligned}$$



Learning the structure of the conditional network automatically, from data.

Conditional sparsification of a multi-layer perceptron

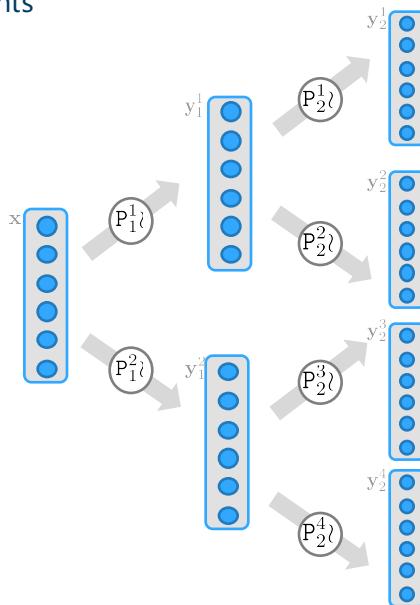
Unzipping the net: results



Training a conditional network via back-propagation

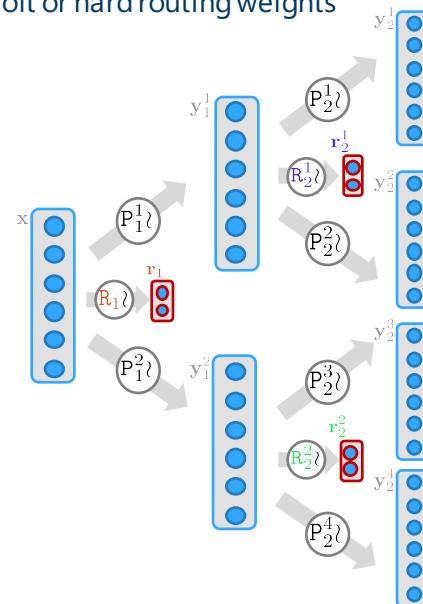
Implicit routing

- Routes do not represent super-classes
- Routers trained with pure back-propagation, for class discriminability.
- Using soft weights



Explicit routing

- Routes represent sets of classes (super-classes)
- Routers trained with priors to "group" similar classes together
- Using soft or hard routing weights



Training in practice

1. We train an implicit routed conditional network, and
2. Train explicit routers afterwards.

This seems to work better in practice than training everything globally via back-prop.