BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks

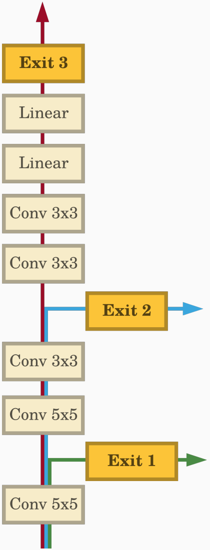
ICPR 2016

Surat Teerapittayanon
Brad McDanel
H. T. Kung

Harvard John A. Paulson School of Engineering and Applied Sciences

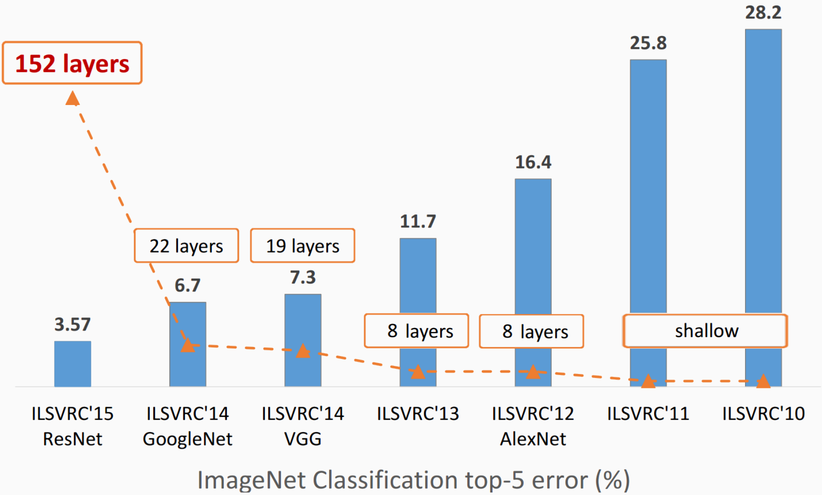
OUTLINE

- Motivation and Background
  - Trend towards deeper networks
  - Auxiliary network structures (GoogLeNet)
- BranchyNet
  - Architecture
  - Training
  - Inference
- Experimental Results
- Future Work
- Conclusion



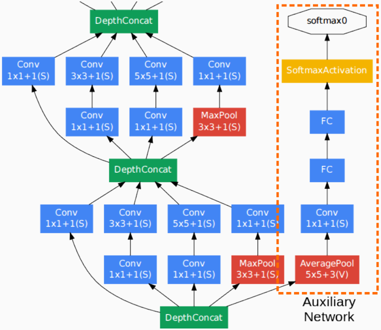
BranchyNet with 3 exits

TREND TOWARDS DEEPER NETWORKS



Accuracy vs. Depth (ILSVRC workshop - Kaiming He)

AUXILIARY NETWORKS

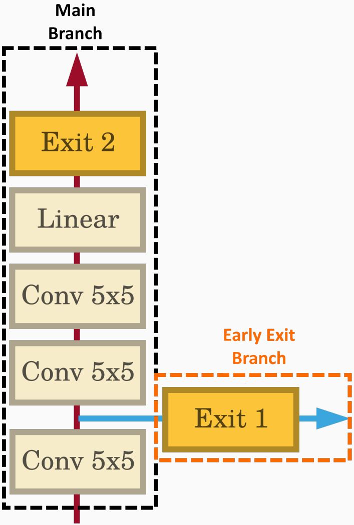


Section of GoogLeNet

- GoogLeNet introduces auxiliary networks
  - Provide regularization to main network
  - Improves accuracy $\approx$ 1%
  - **Removed after training**
  - Only main network is used during inference
- Can we leverage auxiliary networks to address inference runtime of deeper networks?

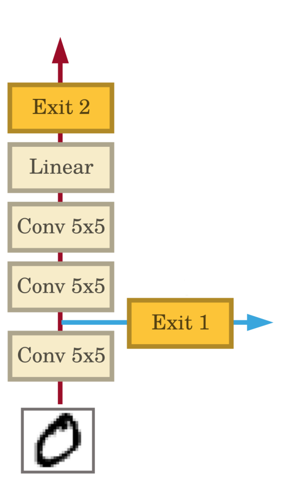
BRANCHYNET

- Easier input samples require lower level features for correct classification
- Harder input samples require higher level features
- Use early exit branches (auxiliary networks) to classify easier samples
  - No computation performed at higher layers
- Requires mechanism for determining network confidence about a sample to use exit
- Jointly training the main and early exit branches improves the quality of lower branches
  - Allowing more samples to exit at earlier points



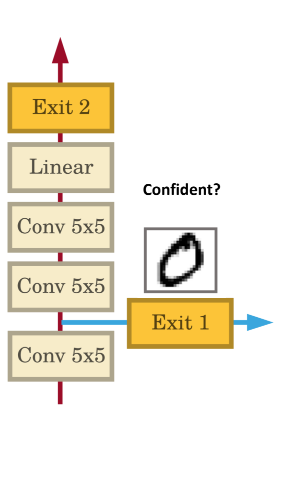
BranchyNet (LeNet)

- New sample enters the network



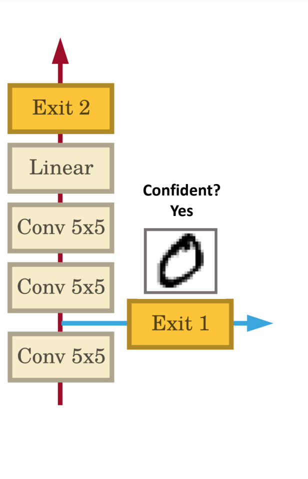
BRANCHYNET EXAMPLE: EASY SAMPLE

- New sample enters the network
- Reaches Exit 1

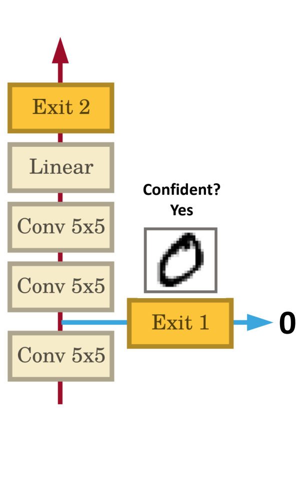


BRANCHYNET EXAMPLE: EASY SAMPLE

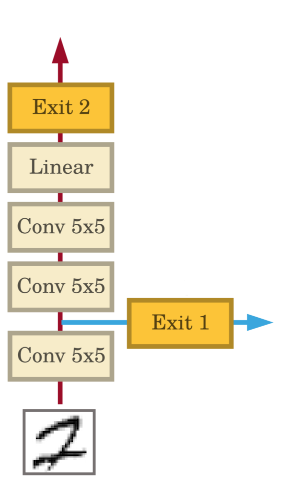
- New sample enters the network
- Reaches Exit 1
- Determined “confident”



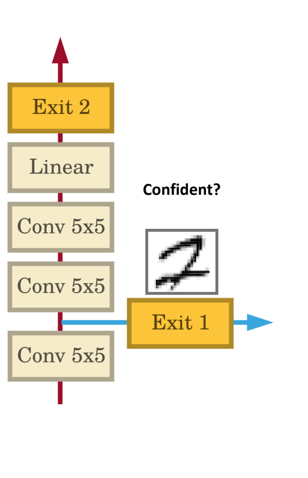
- New sample enters the network
- Reaches Exit 1
- Determined “confident”
- Classifies sample
- No additional work performed at upper layers



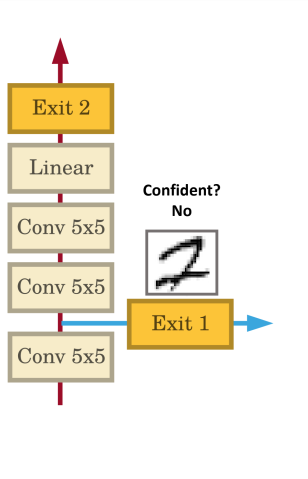
- New sample enters the network



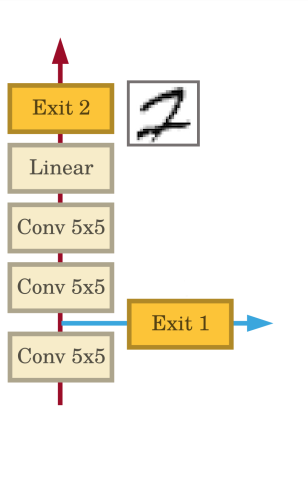
- New sample enters the network
- Reaches Exit 1



- New sample enters the network
- Reaches Exit 1
- Determined “not confident”

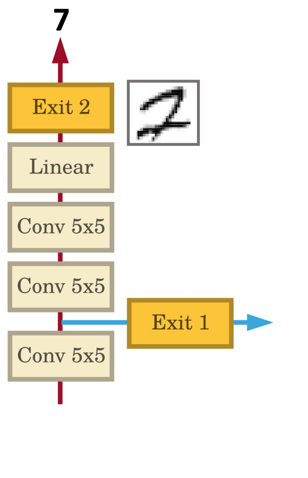


- New sample enters the network
- Reaches Exit 1
- Determined "not confident"
- Continues up the main network (no re-computation of lower layers)



BRANCHYNET EXAMPLE: HARD SAMPLE

- New sample enters the network
- Reaches Exit 1
- Determined "not confident"
- Continues up the main network (no re-computation of lower layers)
- Must exit (classify sample) as Exit 2 is final exit point



MEASURING NETWORK CONFIDENCE

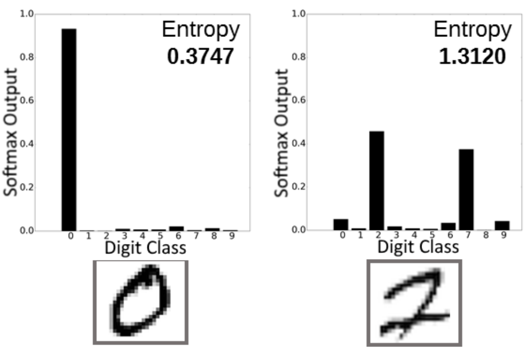
- Use entropy of softmax output to measure confidence

$$\text{entropy}(\mathbf{y}) = \sum_{c \in \mathcal{C}} y_c \log y_c,$$

where $\mathbf{y}$ is a vector containing computed probabilities for all possible class labels and $\mathcal{C}$ is a set of all possible labels

- Choice of entropy versus other measures

Exit 1 Softmax Output



BRANCHYNET TRAINING

- Pretrain main network first
- Add exit branches and train again
- The final loss function is the weighted sum of losses of all exits

$$L_{\text{branchynet}}(\hat{y}, y; \theta) = \sum_{n=1}^{N} w_n L(\hat{y}_{\text{exit}_n}, y; \theta),$$

where N is the total number of exit points

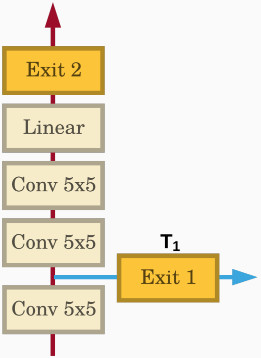
- Early exit weights $W_{1..N-1} = 1$
- Last exit weight $W_N = 0.3$

BRANCHYNET INFERENCE

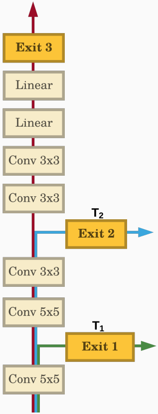
```
procedure BranchyNetFastInference(x, T)
    for n = 1..N do
        z = f_exit_n(x)
        ŷ = softmax(z)
        e = entropy(ŷ)
        if e < T_n then
            return arg max ŷ
    return arg max ŷ
```

Figure: BranchyNet Fast Inference Algorithm. $\mathbf{x}$ is an input sample, $\mathbf{T}$ is a vector where the n-th entry T_n is the threshold for determining whether to exit a sample at the n-th exit point, and N is the number of exit points of the network.

- Network Architectures
  - LeNet (on MNIST)
  - AlexNet (on CIFAR-10)



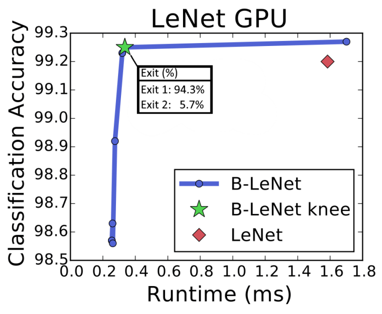
Branchy-LeNet

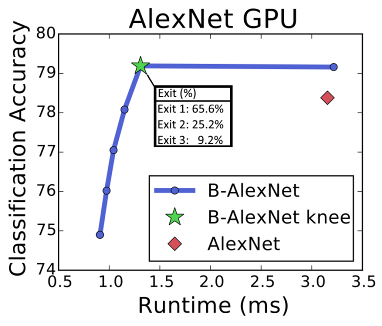


Branchy-AlexNet

RESULTS

- Points on the curve found by sweeping over values of T
  - In the case of more than one early exit, we take combinations of T_i values
- Accuracy improvement over baseline network (red diamond) due to joint training
- Runtime improvements over baseline network due to classifying the majority of samples at early exit points (no computation performed for higher layers)
- As T values increase, more samples exit at the higher exit branches





FUTURE WORK

- Automatically find the threshold values T for each exit branch
- Investigate alternative confidence measures other than softmax entropy (e.g., OpenMax, GANs)
- Dynamically adjusting the weight of loss based on individual samples
    - Easier samples have more weight at lower branches
    - Harder samples have more weight at higher branches

- Introduce a mechanism to exit a percentage of samples at earlier points in the network
- Jointly training these exit points improves accuracy which allows additional samples to exit early
- Achieve a factor of 2-4x speedup compared to baseline single network for our test case
- BranchyNet implementation written in Chainer and open source: `https://gitlab.com/htkung/branchynet`

Thanks for your attention!
Comments and Questions?

RESULTS TABLE

Table: Selected performance results for BranchyNet on the different network structures. The BrachyNet rows correspond to the knee points (denoted as green stars in the previous slides).

	Network	**Acc.** (%)	**Time** (ms)	**Gain**	**Thrshld. T**	**Exit** (%)
CPU	LeNet	99.20	3.37	-	-	-
	B-LeNet	99.25	0.62	5.4x	0.025	94.3, 5.63
	AlexNet	78.38	9.56	-	-	-
	B-AlexNet	79.19	6.32	1.5x	0.0001, 0.05	65.6, 25.2, 9.2
GPU	LeNet	99.20	1.58	-	-	-
	B-LeNet	99.25	0.34	4.7x	0.025	94.3, 5.63
	AlexNet	78.38	3.15	-	-	-
	B-AlexNet	79.19	1.30	2.4x	0.0001, 0.05	65.6, 25.2, 9.2