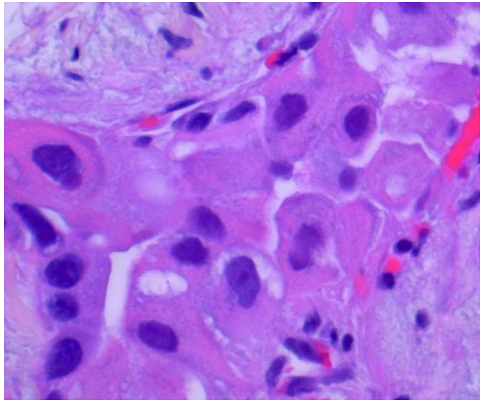
Lung Cancer Detection with Convolutional Neural Networks

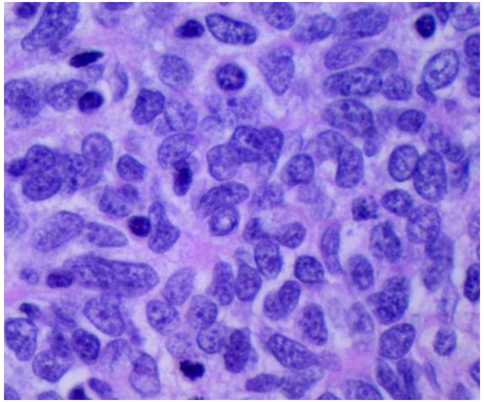
Abuduani Niyazi, Derek Kielty, Rishat Dilmurat, Sujoy Upadhyay, Ronak Desai

Introduction

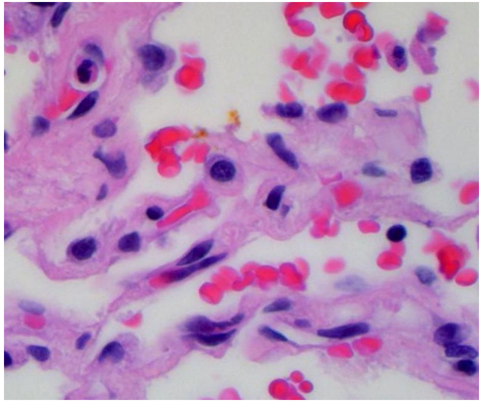
Trained convolutional neural networks (CNNs) to classify microscope images of lung cells as cancerous (2 types) or non-cancerous



cancerous
(adenocarcinoma)



cancerous
(squamous cell carcinoma)



non-cancerous
(normal lung cells)

Introduction

- Dataset: “Lung and Colon Histopathological Images” (kaggle.com)
- 15,000 RGB images (5000 of each cell type) of size 768x768
- Constructed from 250 original images (of each cell type) then augmented to 5000 by geometric transformations (rotations, reflections, shears, etc.)

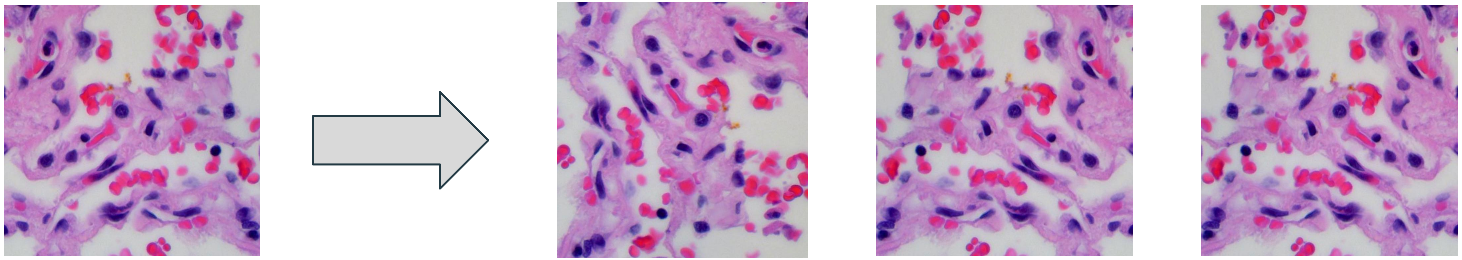
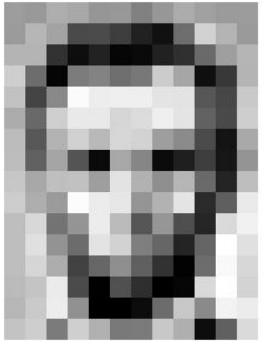
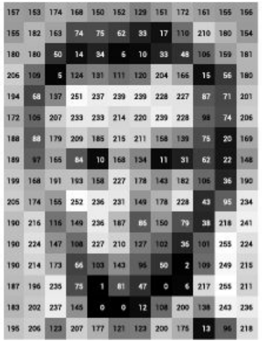


Image = Matrix

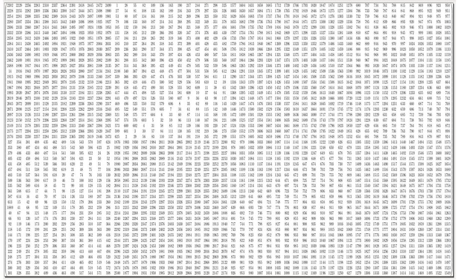
Image => **digital image =>** **Matrix**

NOWADAYS: 5,000 x 5,000 pixels (or more) for larger ones

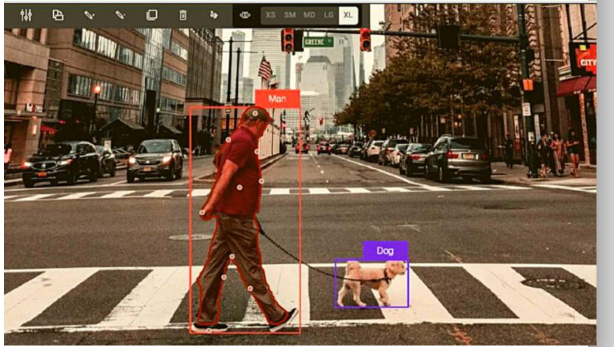




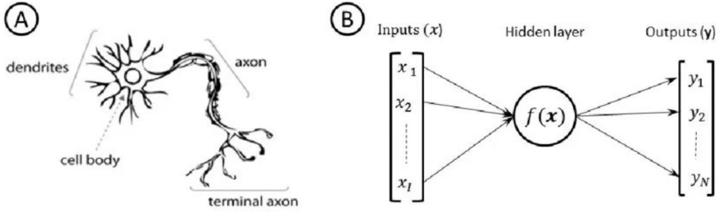
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218



Our aim is to recognize objects in images as quickly and efficiently as possible.

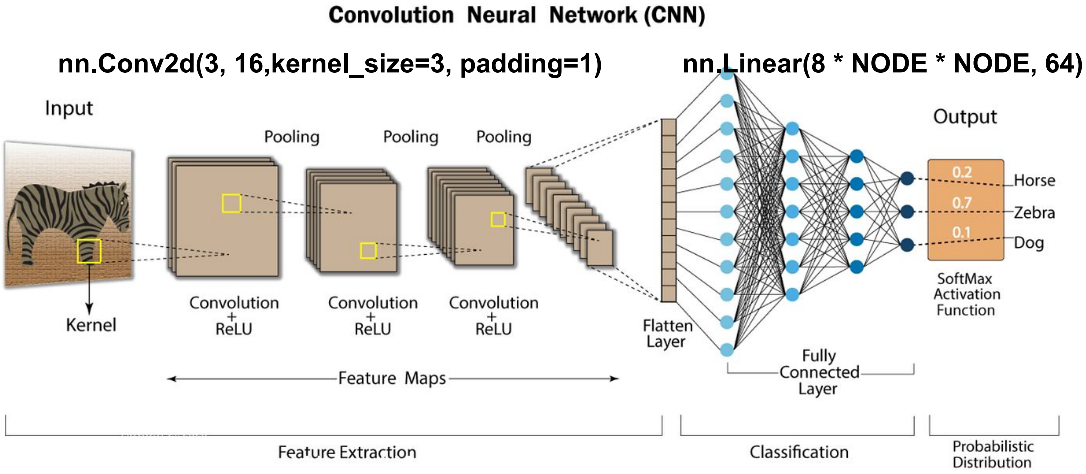


CNN - Convolutional Neural Network



1. Input layer
2. Convolutional Layers
3. Activation Functions
4. Pooling Layers
5. Fully Connected Layers
6. Output Layer

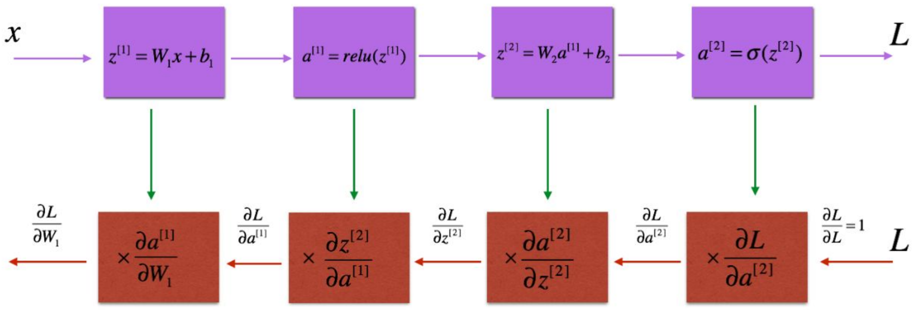
nn.Conv2d(3, 16,kernel_size=3, padding=1) **nn.Linear(8 * NODE * NODE, 64)**



Pytorch code includes **data loading**, **model definition**, **training**, and **evaluation**.

Convolution layer	Fully connected layer
self.conv1 = nn.Conv2d(3, 16, kernel_size=3, padding=1)	self.fc1 = nn.Linear(8 * NODE * NODE, 64)
self.act1 = nn.Tanh()	self.act3 = nn.ReLU()
self.pool1 = nn.MaxPool2d(2)	self.fc2 = nn.Linear(64, 3)

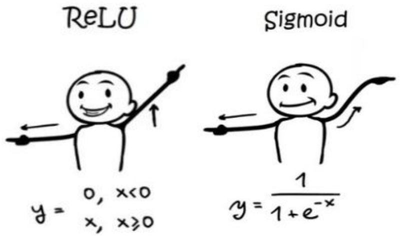
Forward and backward propagation



Forward: Z = WX + b

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial a^{[1]}} \frac{\partial a^{[1]}}{\partial W_1}$$

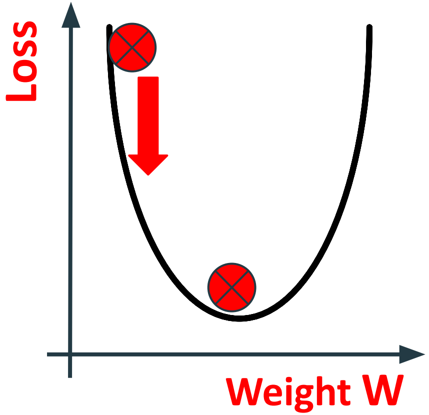
Activation Function



Backward: **Gradient descent**

Repeat {
W = W - algha*d(loss)/d(W)

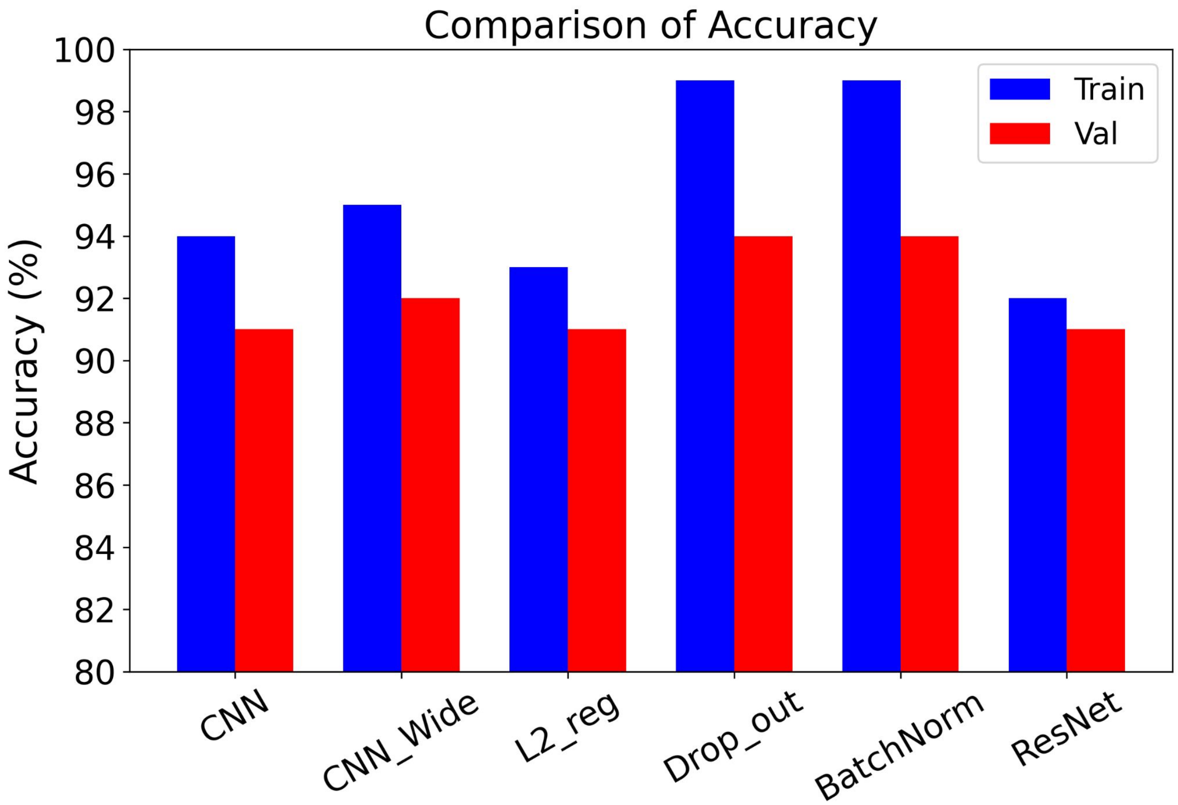
b = b - algha*d(loss)/d(b)
}

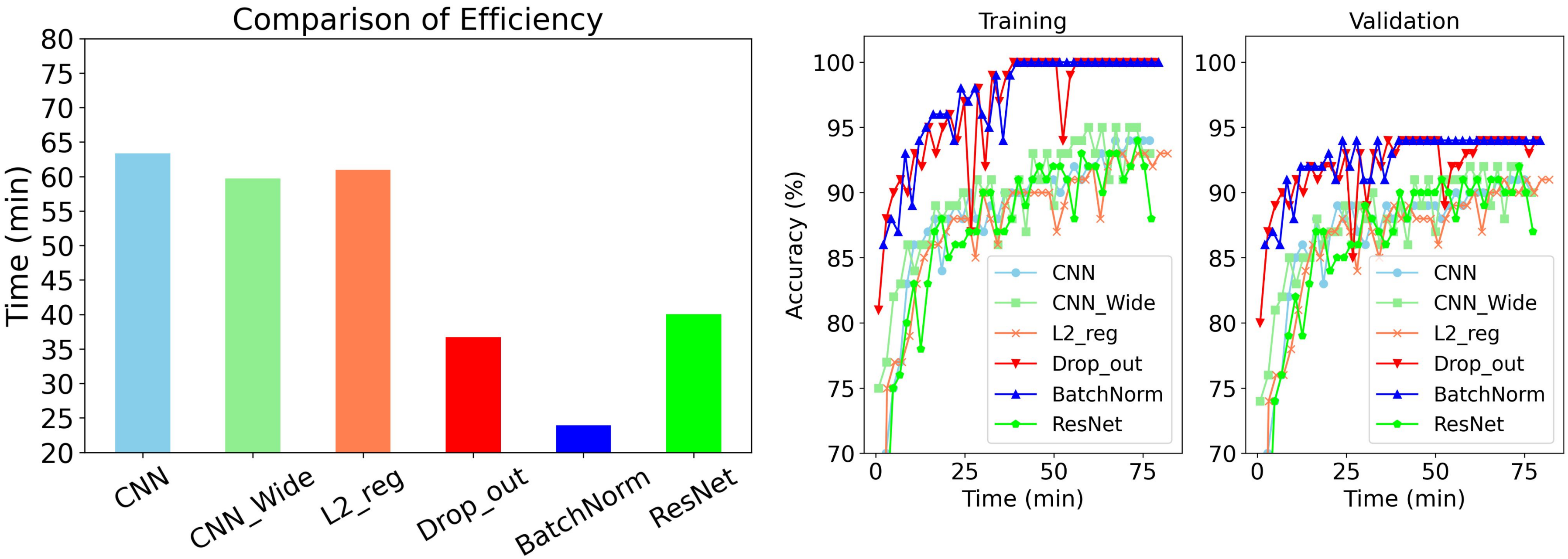


- Improvement on Baseline CNN model with different methods.

- Dropout and batch normalization provide **best improvement**.

- Single Block ResNet gives **no improvement** over Baseline CNN.





- Batch Normalization, Drop out and ResNet **improve the training efficiency**.

- Batch Normalization **improves training accuracy** and **improves the training efficiency by large margin**.

Some Advanced Models

- Channel Boosted CNN: [1804.08528] A New Channel Boosted Convolutional Neural Network using Transfer Learning (arxiv.org)
    - Leverages additional channels from pre-trained networks
    - We are not using pre-trained networks, so this may not be beneficial
- Residual Network (ResNet): [1512.03385] Deep Residual Learning for Image Recognition (arxiv.org)
    - Introduced “Skip” Connections which allow training of very deep networks
    - We attempt a 20 and 40 layer network (c.f. baseline of only 6 layers)

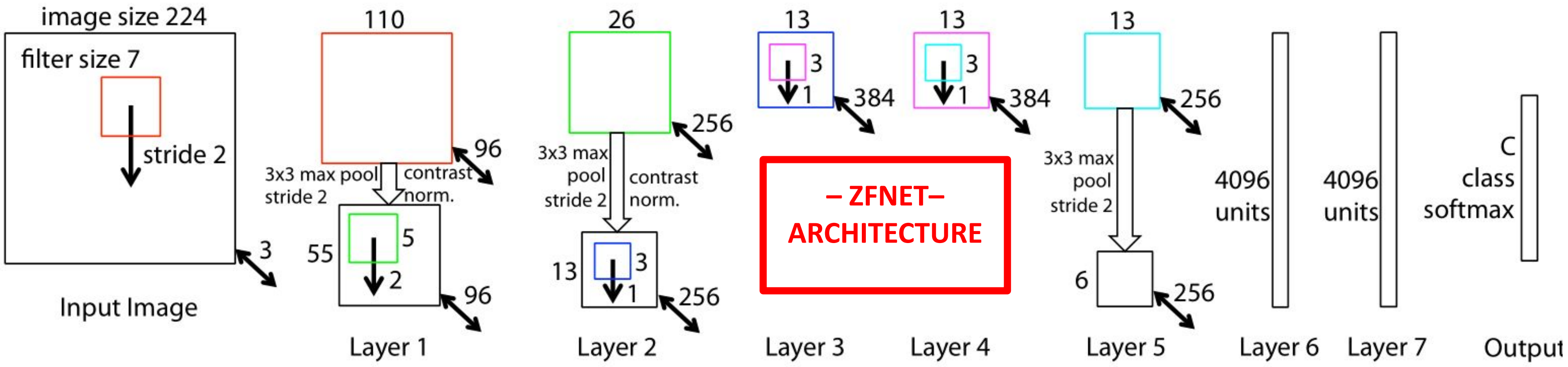
AlexNet and ZFNet

AlexNet: [ImageNet classification with deep convolutional neural networks | Communications of the ACM]

- Won ImageNet Large Scale Visual Recognition Challenge in 2012

ZFNet: [Visualizing and Understanding Convolutional Networks | SpringerLink]

- Similar Architecture, but improved to win ImageNet 2013 Challenge
- 5 Convolutional Layers, 3 Dense Layers, Max pooling, **Dropout**

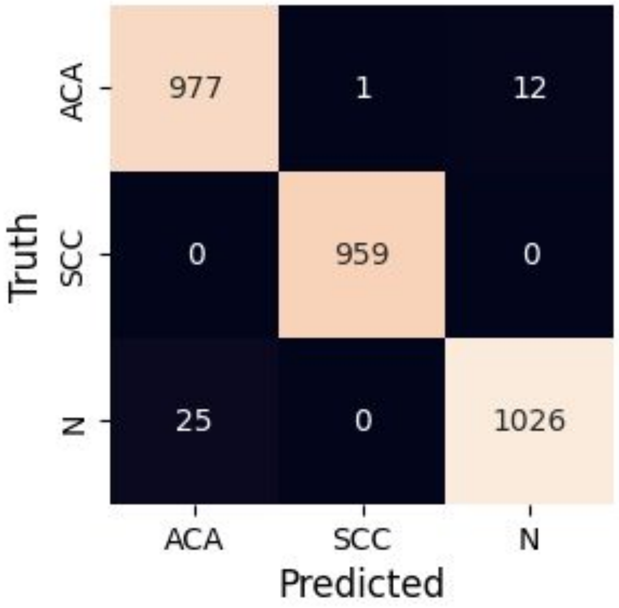


– ZFNET– ARCHITECTURE

ZFNet Results

- **High Accuracy**
  - 99.5% Accuracy on Training Set
  - 98.8% Accuracy on Validation Set

- **Confusion Matrix: Visualize Misclassifications**
  - No confusion between SCC and N
  - Some confusion between ACA and N



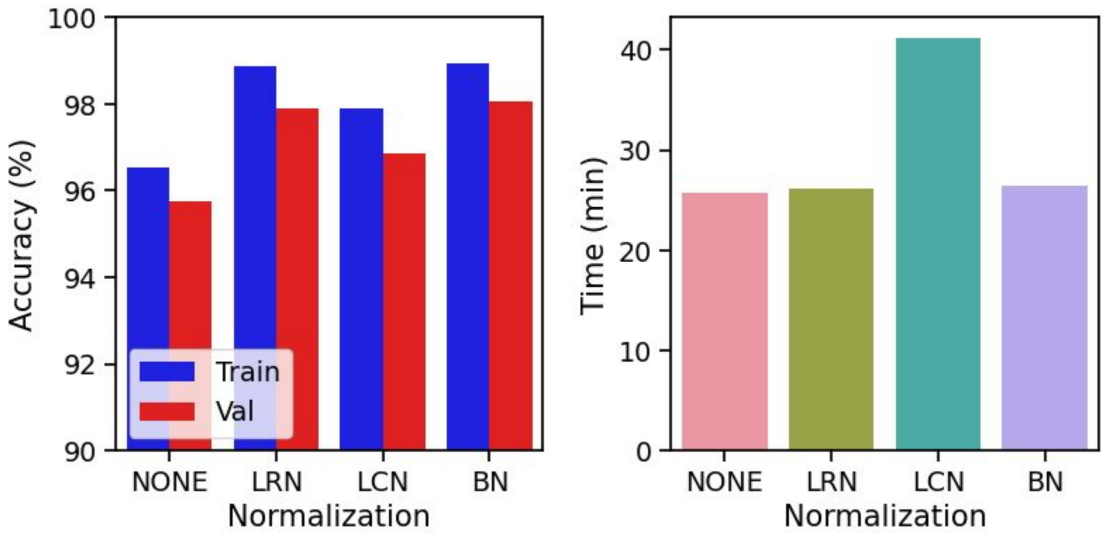
ZFNet Experiments

Local Response Normalization (LRN) used in AlexNet (2012)

Local Contrast Normalization (LCN) used in ZFnet (2013)

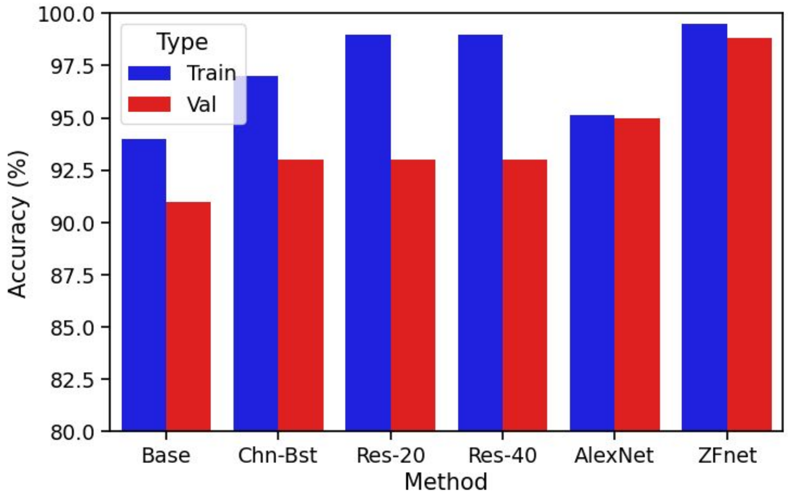
Batch Normalization (BN) first used in (2015)

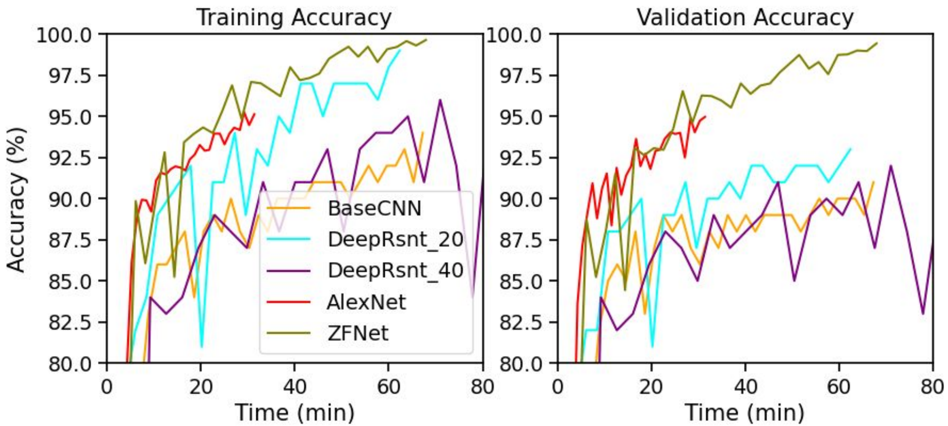
- All normalization methods provide **better performance** than without.
- Normalizations **do not** appreciably **increase the time to train**
- *Exception*: LCN
  - No built in pytorch method
  - Custom implementation may be inefficient



Best Models

- **ZFNet** architecture had the highest accuracy by far.
    - > 99 % accuracy
- **Deep Resnet** with 40 layers **takes significantly longer to train** than other models (> 80 min)
    - More complex isn't always better
- AlexNet accuracy is good but didn't train long enough
    - Validation accuracy did not plateau





Conclusion

- We used an existing Kaggle Dataset as a testbed for various CNN techniques
- We tweaked parameters of the CNN to understand what works best on our data
    - Found that more complex models with wider or deeper architectures are not always better
- We manually implemented some advanced CNN architectures that have good performance on past data sets
- We achieved high performance with one model: ZFNet with over 99% accuracy on the dataset