REGULARIZATION STRATEGIES IN DEEP LEARNING

CSE 676 – Project 2

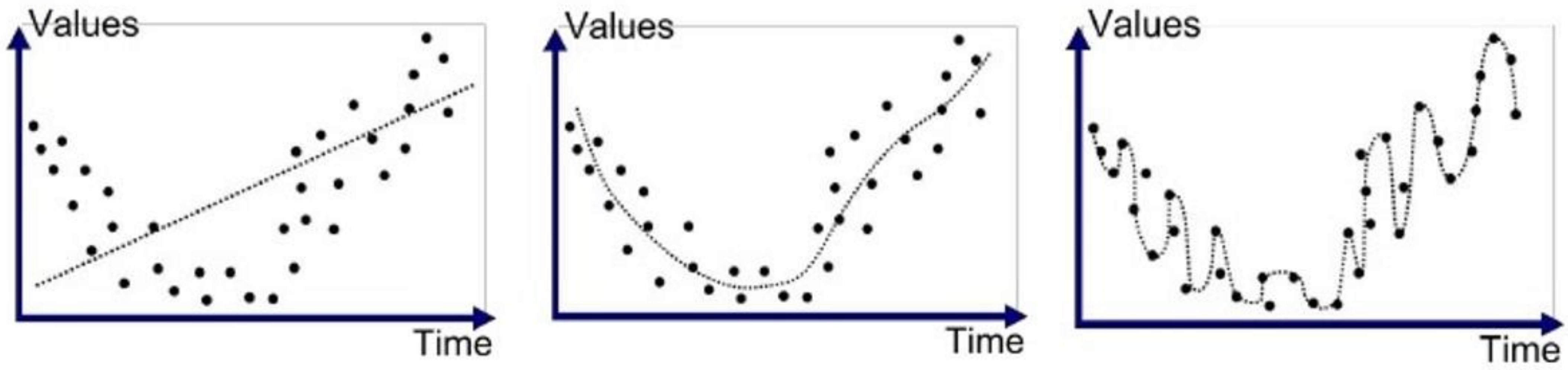
-Sandesh Kumar Srivastava

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Underfit vs Good Fit vs Overfit

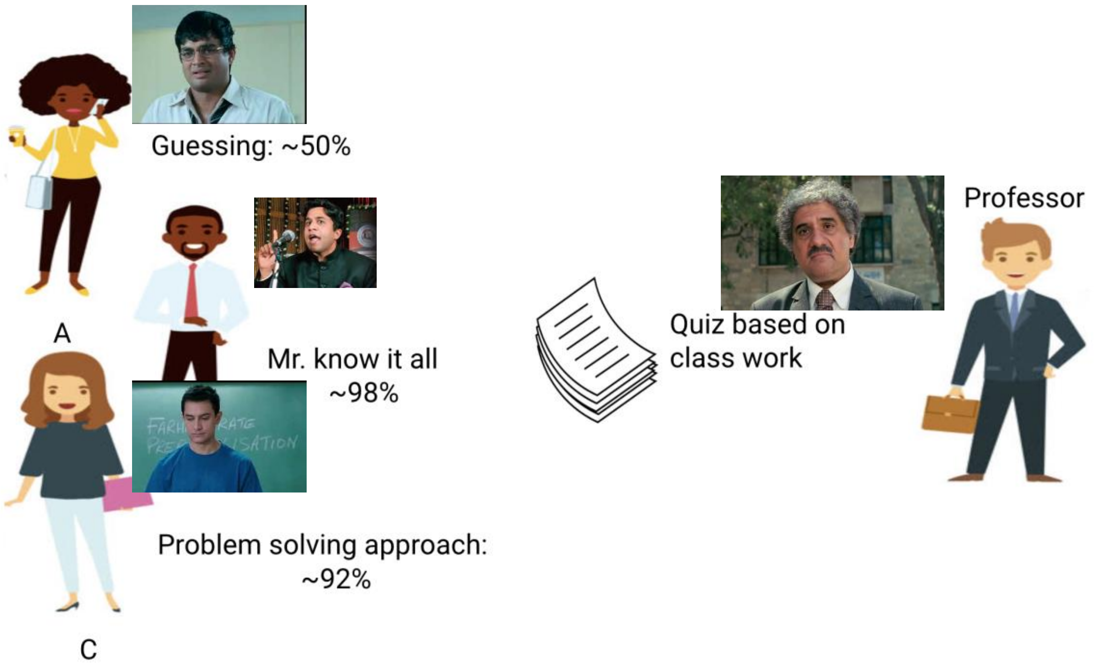
- Underfit or High Bias **Solution: Make model more expressive/complex**
- Good fit or low bias and low variance
- Overfit or High Variance **Solution: Regularization**

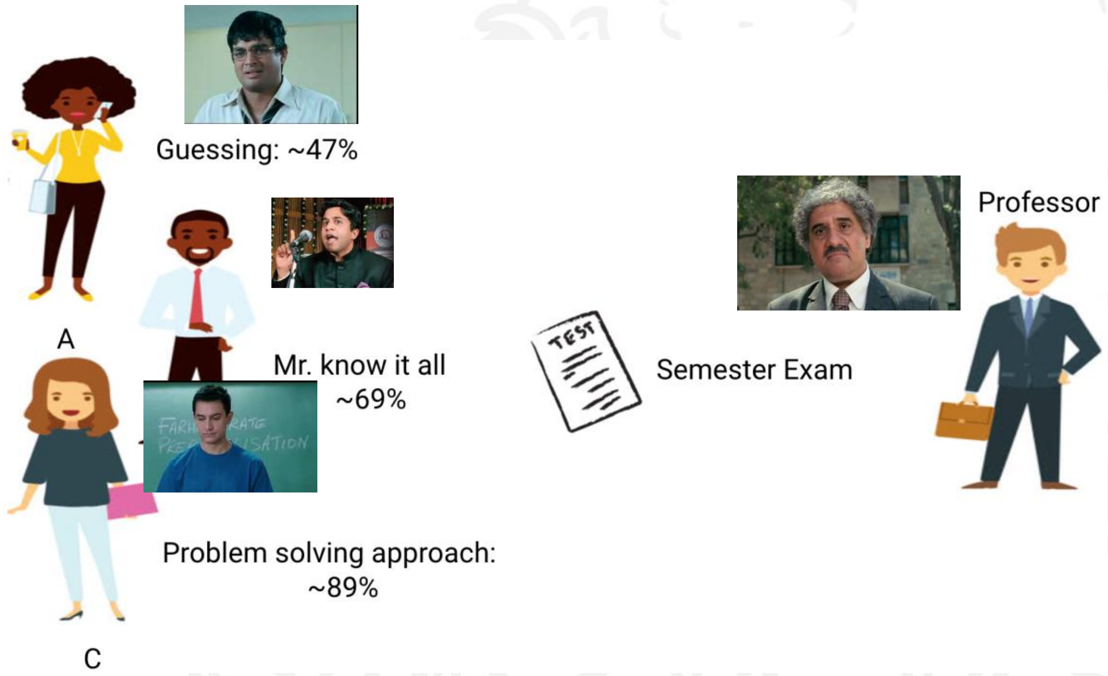
John von Neumann famously said "With four parameters I can fit an elephant, and with five I can make him wiggle his trunk."



Underfitted   Good Fit/Robust   Overfitted

Overfitting in real life example





Regularization Strategies

Methods to reduce generalization error(error on unseen data)

-> But not the training error

-> Even at the expense of training error

Different methods exist:

- L1 norm
- L2 norm
- Data Set Augmentation
- Noise Robustness
- Early Stopping
- Dropout
- Adversarial Training ..

Regularization Strategies

L1 norm:

$$\tilde{J}(\Theta; X, y) = J(\Theta; X, y) + \alpha\Omega(\Theta) \quad (1)$$

$$\Omega(\Theta) = ||w||_1 = \sum |w_i|_1 \quad (2)$$

$$\tilde{J}(\Theta; X, y) = J(\Theta; X, y) + \alpha \sum_i |w_i|_1 \quad (3)$$

```
Conv2D(filters=f_5x5, kernel_size=(5,5),
padding='same', activation='relu', kernel
_regularizer=l1(l1=0.01), bias_regularize
r=l1(1e-4), activity_regularizer=l1(1e-
5))
```

L2 norm:

$$\tilde{J}(\Theta; X, y) = J(\Theta; X, y) + \alpha\Omega(\Theta) \quad (1)$$

$$\Omega(\Theta) = \frac{1}{2}||w||_2^2 \quad (4)$$

$$\tilde{J}(\Theta; X, y) = J(\Theta; X, y) + \alpha\frac{1}{2}||w||_2^2 \quad (5)$$

```
Conv2D(filters=f_3x3_r, kernel_size=(1,1)
, padding='same', activation='relu', kern
el_regularizer=l2(l2=0.001), bias_regular
izer=l2(1e-4),
activity_regularizer=l2(1e-5))
```

Regularization Strategies

Data Set Augmentation:

- Train the ML model on more data
- Transform the given input to obtain new input

```
datagen = ImageDataGenerator(
    zca_epsilon=1e-06,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True)
```

Noise Robustness:

- Noise can be applied at different levels to a ML model.
- If applied at input, it serves as a data augmentation.
- If applied to output layers, it helps to handle the mistakes made by ML model.

```
GaussianNoise(0.005)
```

Regularization Strategies

Early Stopping:

- Stop training process whenever there is not significant improvement on the validation data metrics

```
EarlyStopping(monitor='val_accuracy', patience=40)
```

Dropout:

- Technique similar to bagging.
- Randomly dropping some units by simply multiplying their output value to 0.

```
Dropout(0.5)
```

Adversarial Training:

- ML model trained on the generated adversarial examples.
- Fast Gradient Sign Method(FGSM)

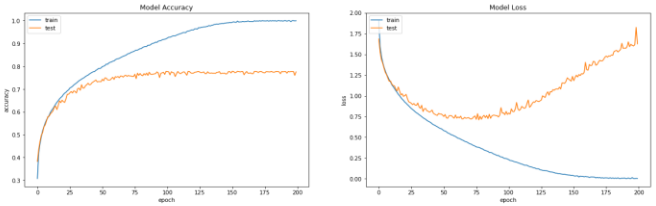
$$x \rightarrow x + \epsilon sign(\nabla_x J(\Theta, x, y)) \quad (6)$$

```
gradient = tape.gradient(loss, image)

signed_grad = tf.sign(gradient)

adversarial = image + perturbations * epsilon
```

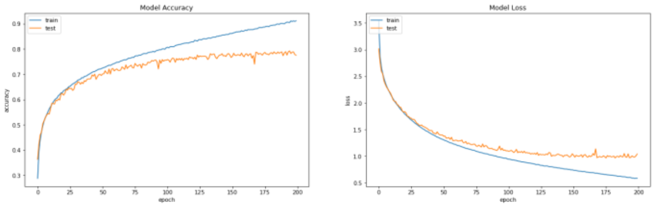
Implementation

- Using Keras library in Python.
- CIFAR-10 dataset(50k training and 10k test images of size 32x32x3).
- Inception_v2 like model.
- Adamax optimizer.
- ModelCheckPoint to store best weights.

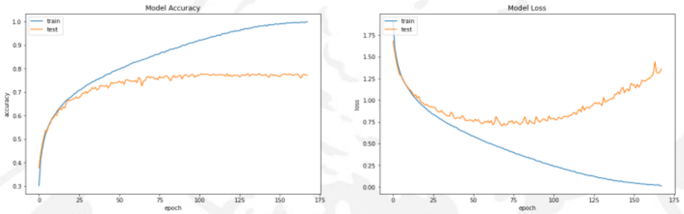
Results - accuracy and loss plots



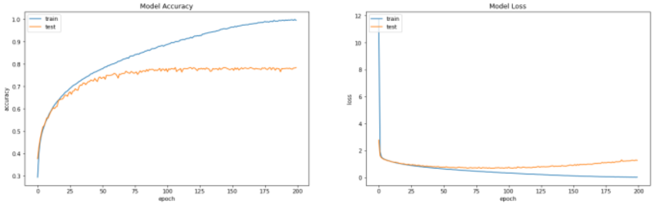
No Regularization



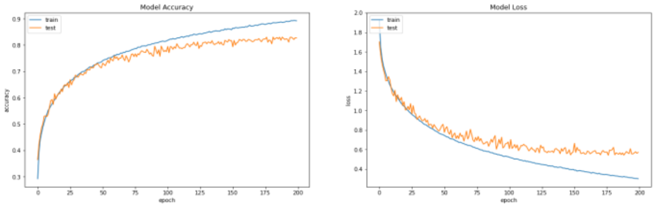
L1L2 norm



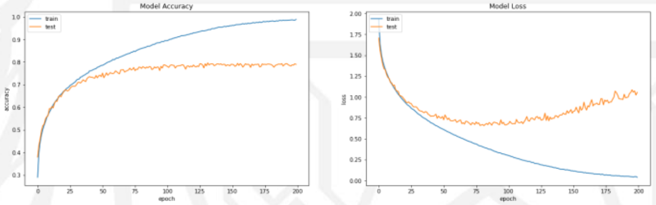
Early Stopping



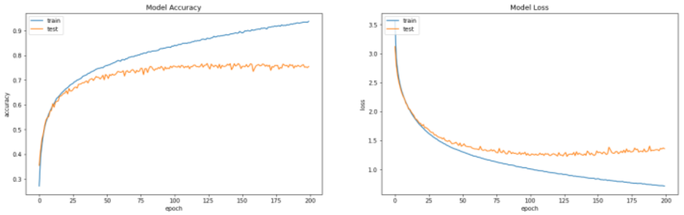
L1 norm



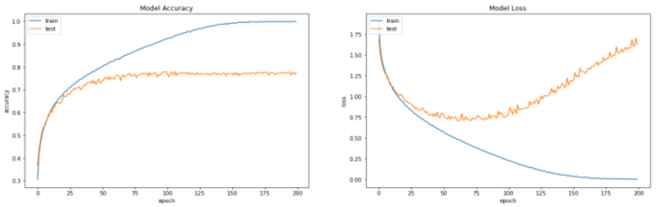
Dataset Augmentation



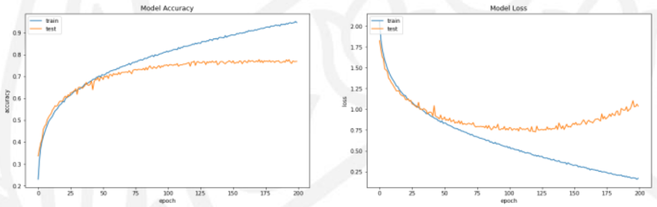
Dropout



L2 norm



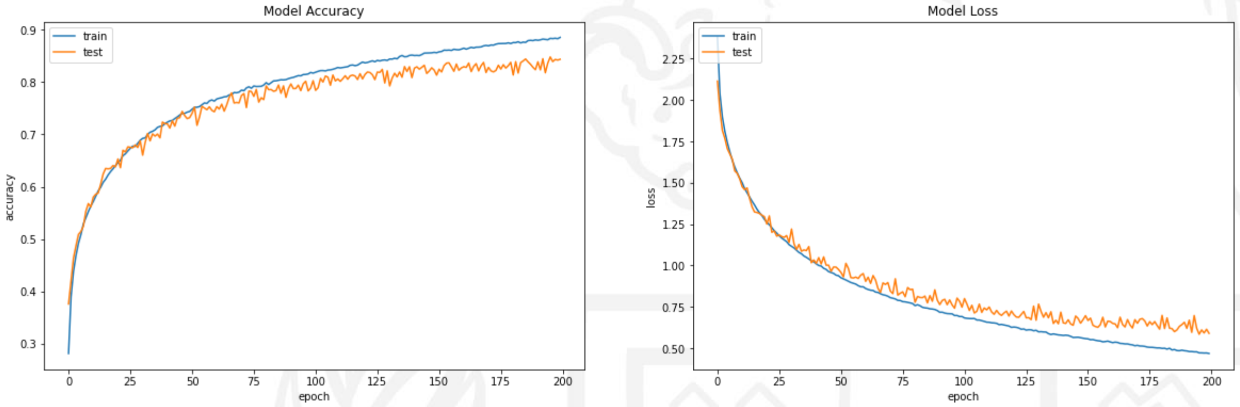
Noise Robustness



Adversarial Training(FGSM)

Results

Regularization Strategy	Precision	Accuracy
No regularization	0.77896	0.7785
L1	0.78399	0.7841
L2	0.79902	0.8005
L1L2	0.79272	0.7923
Dataset Augmentation	0.82933	0.8301
Noise Robustness	0.78409	0.7811
Early Stopping	0.77789	0.7703
Dropout	0.79751	0.7961
Adversarial(FGSM)	0.77787	0.777
Combination	0.84854	0.8481



Combination
(L2, Dataset Augmentation & Dropout)

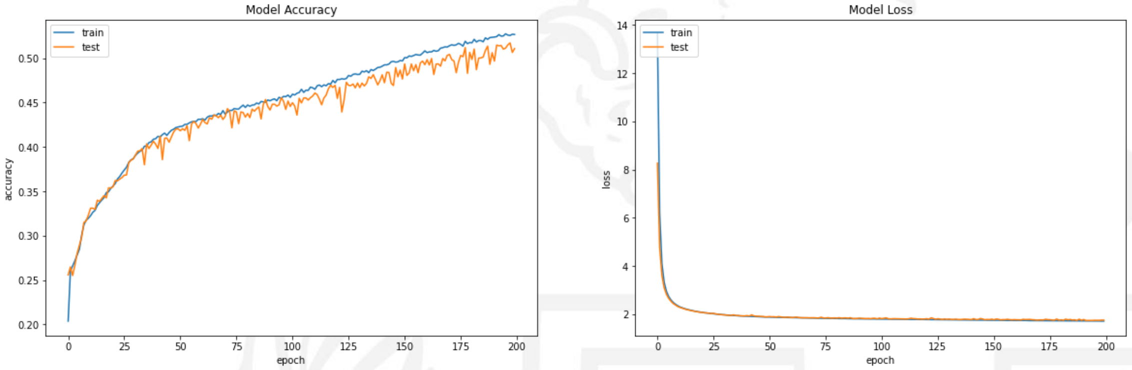
Achieved in 168 epochs compared to 200 for others

Accuracy on adversarial test samples increased from 34% to 70%

Best performance

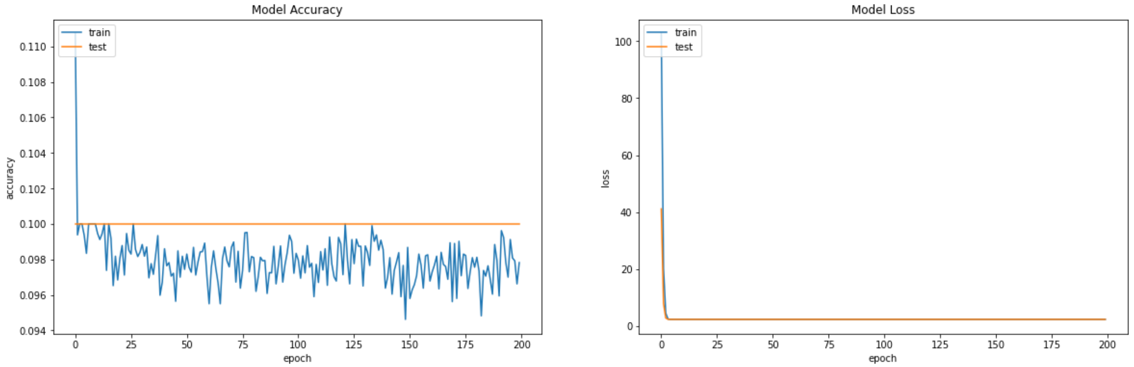
Results – finetuning regularization hyperparameters

L2 kernel regularizer	Precision	Accuracy
1e-1	0.01	0.1
1e-2	0.504229	0.5173
1e-3	0.79902	0.8005
1e-4	0.77536	0.7759

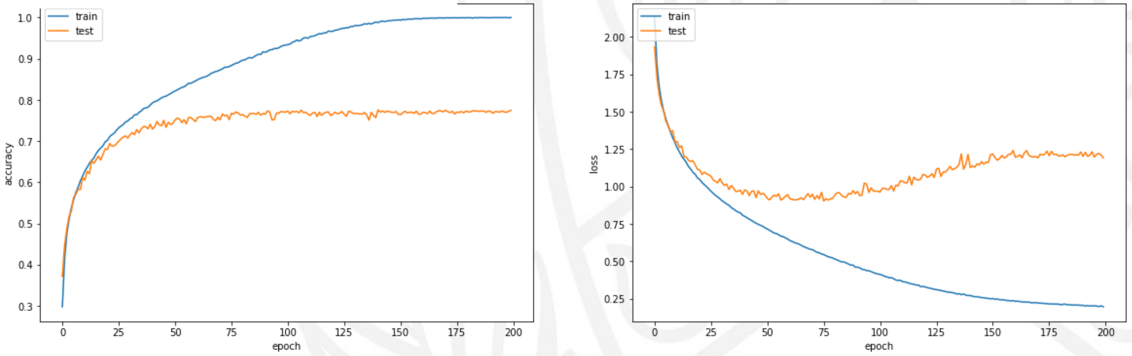


1e-2

$$\tilde{J}(\Theta; X, y) = J(\Theta; X, y) + \alpha\Omega(\Theta) \qquad (1)$$



1e-1



1e-4

Conclusion

- Overfitting is a very common problem in deep neural networks.
- Regularization strategies help to solve the problem of overfitting.
- Selection of regularization strategy depends on the nature of problem being solved.
- Often multiple regularization strategies are combined to get best result.
- Finetuning of regularization parameters is required for optimal performance.