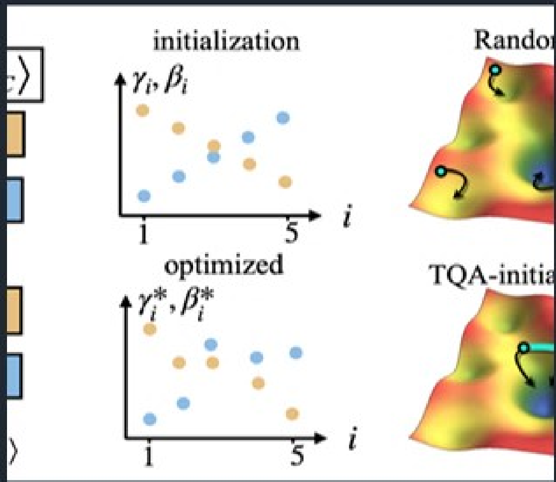
Optimization Practices in Modern AI



Essentials:

Theory and hands-on coding

Pre-req

Good exposure to ML/DL

Comfortable with sklearn, Pytorch, TF/Keras

Familiar with Images, text and time series data

Optimization

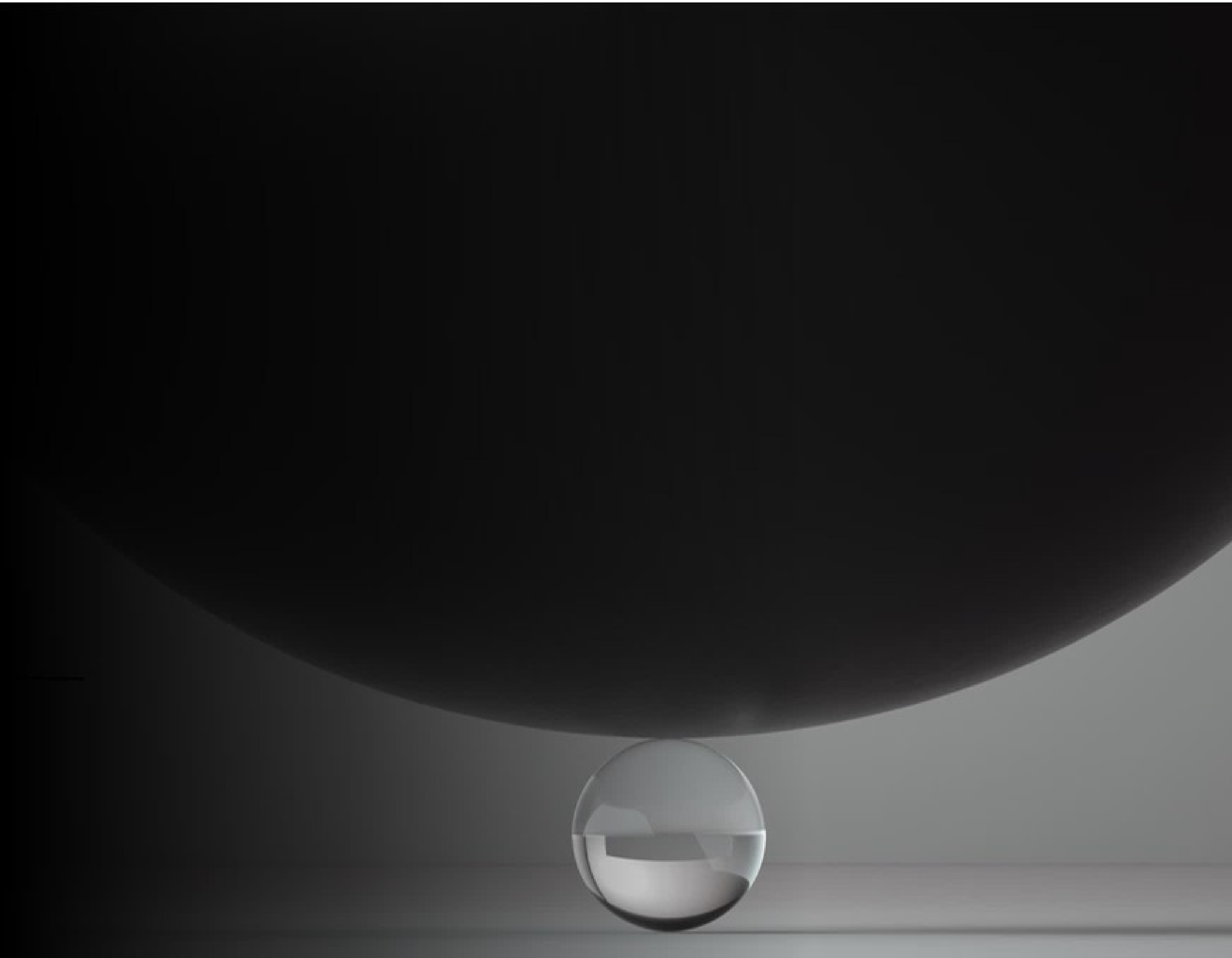
Optimization techniques

- Explanation of L1 and L2 regularization techniques.
- Dropout
- Cross-Validation
- Differential Learning Rates
- Convergence Monitoring and Early Stopping
- batch normalization
- Data Augmentation

Optimizers

- Understanding Optimization Challenges:
- Momentum
- Adam optimizer
- RMSProp
- Adadelta
- Adagrad
- Vanishing & Exploding gradients
- Optimizers for Specific Architectures (CNN)
- Optimizers for Specific Architectures (RNN)

Dropout



Definition of Dropout

- is a regularization technique used in deep learning to prevent overfitting and improve the generalization of neural networks.
- works by randomly "dropping out" a fraction of the neurons during the training phase.
- means that during each forward and backward pass, some neurons are randomly selected and ignored, effectively setting their output to zero.

How Dropout Works

Training

- During each iteration of training, each neuron has a probability p (the dropout rate) of being set to zero.
- neurons that are dropped out do not contribute to the forward pass and do not participate in backpropagation.
- creates different "thinned" networks with each iteration, where the network can be seen as an ensemble of many smaller networks.

Testing

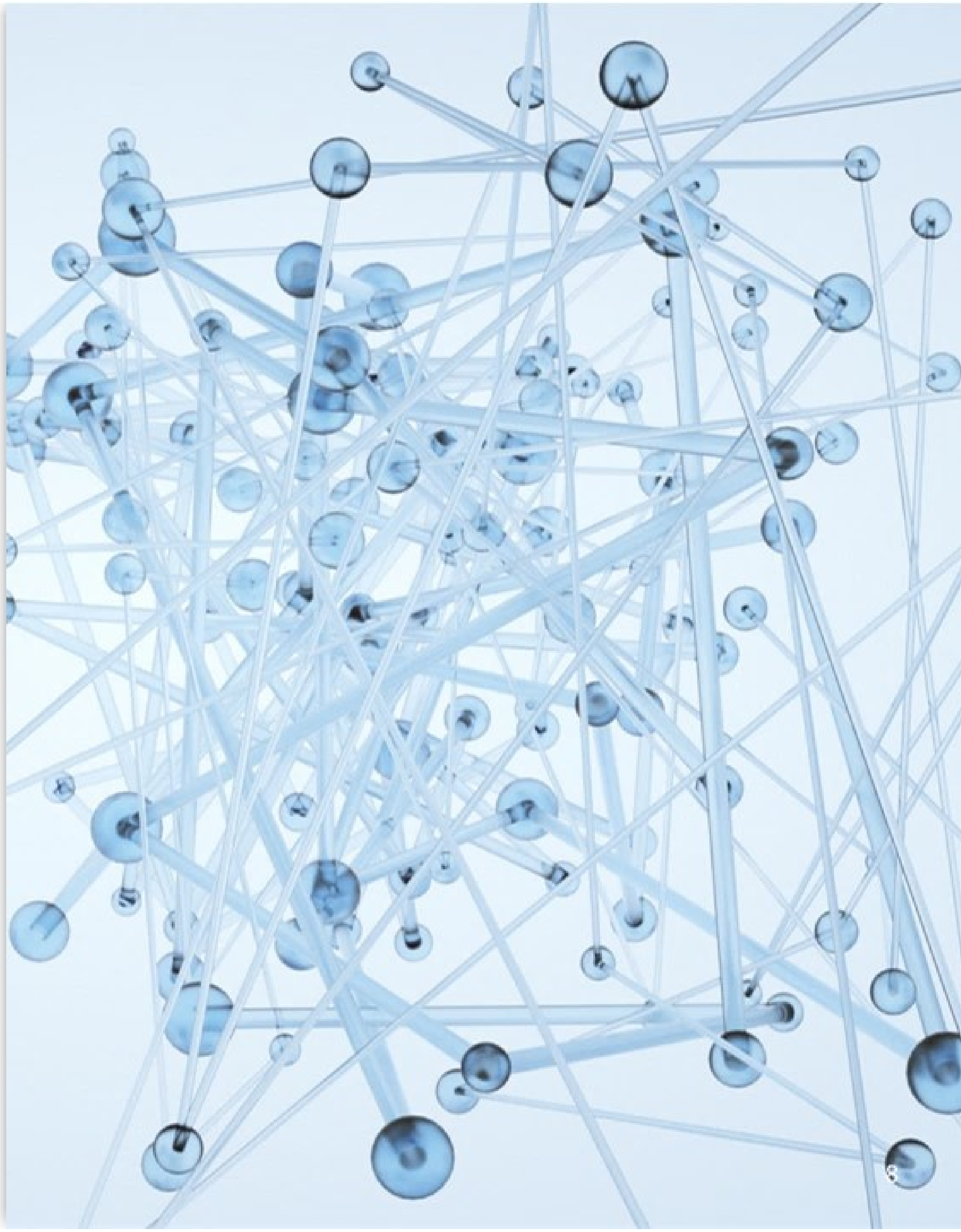
- During testing, no neurons are dropped out.
- weights of the neurons are scaled down by the dropout rate p (i.e., multiplied by $1-p$).
- scaling ensures that the output at test time is the average of the outputs of the exponentially many thinned networks.

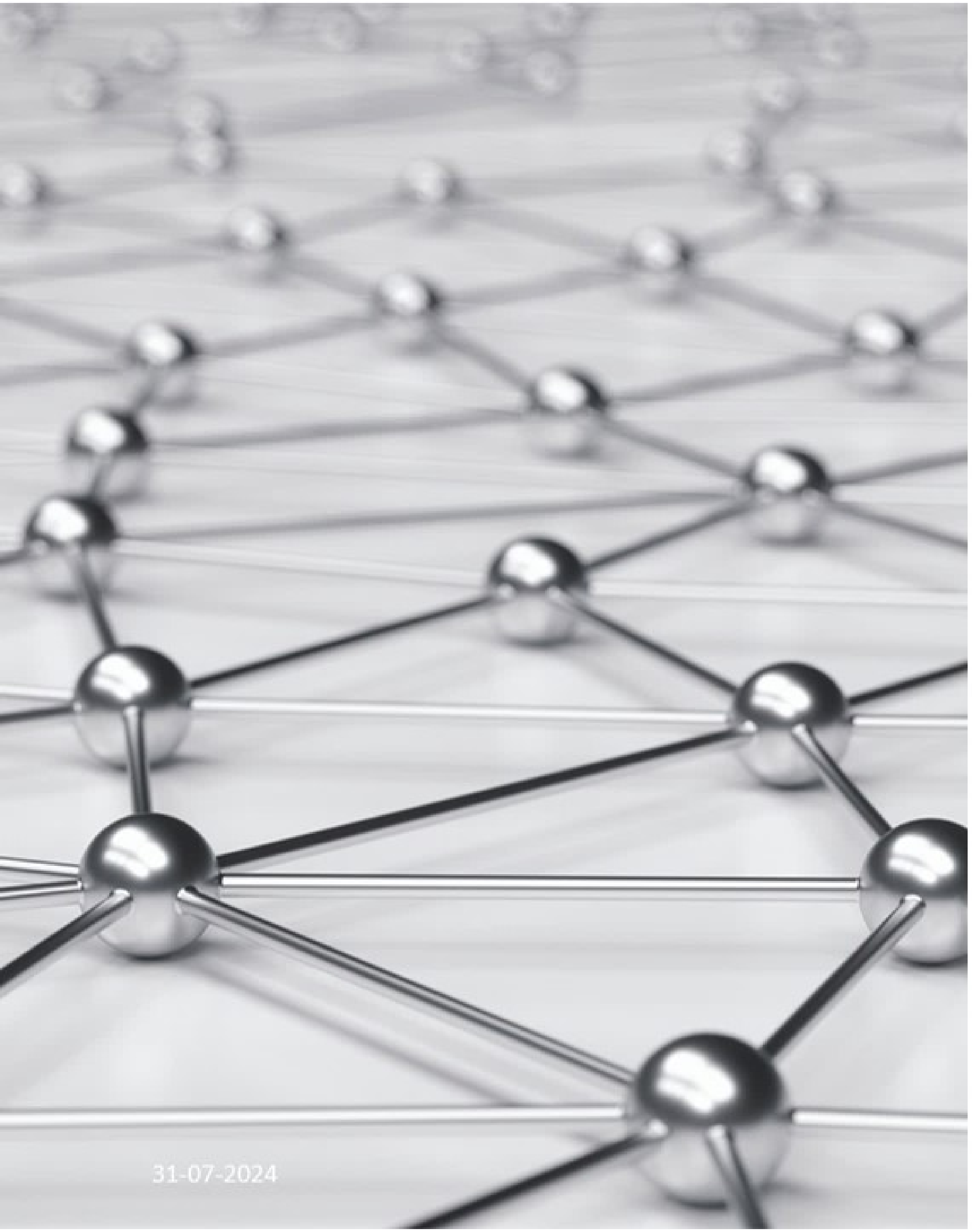
Visualization of the Concept

- Consider a simple network with 5 neurons in a layer:
  - **Iteration 1**: Dropout randomly deactivates neurons 2 and 4.
  - **Iteration 2**: Dropout randomly deactivates neurons 1 and 3.
  - **Iteration 3**: Dropout randomly deactivates neurons 2 and 5.
- And so on...
- Each iteration presents a different configuration of active and inactive neurons.
- Over many iterations, the network learns to perform well regardless of which subset of neurons is active.

Ensemble Interpretation

- During training, we have many "thinned" networks, each with a different architecture due to the random dropout.
- Each of these "thinned" networks contributes to the learning process by updating the weights in a way that is beneficial across many configurations.
- full network, with all neurons active during testing, combines the knowledge gained from these various smaller networks.





Purpose of Dropout

- Preventing <u>Overfitting</u>
  - By randomly dropping out neurons, dropout <u>prevents</u> the network from becoming <u>too reliant on specific neurons</u>
  - makes the network more robust and less likely to overfit the training data.
- Thus, Improving <u>Generalization</u>
  - Since neurons cannot rely on the presence of specific other neurons during training, they must learn features that are generally useful.

Mechanism of Dropout

Neurons are randomly deactivated

Reduces co-adaptation

Creates an ensemble effect

Practical Use of Dropout

Dropout is typically applied to the fully connected layers of a neural network.

Common dropout rates are between 0.2 to 0.5.

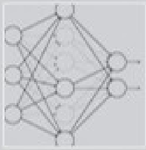
Using a rate of 0.5 means that half of the neurons are dropped out on average during each iteration.

Dropout can be easily implemented in most deep learning frameworks (such as TensorFlow, Keras, and PyTorch) using built-in functions.

Demo using python/sklearn

(Example of Dropout in Code)

Dropout Probability



Dropout Rate (p): fraction of neurons that are randomly set to zero in each training iteration.

For example, if the dropout rate is 0.5, then on average, half of the neurons are dropped out during each iteration.

Keep Probability ($1-p$): probability that a neuron remains active during training.

For instance, if the dropout rate is 0.5, the keep probability is also 0.5, meaning there's a 50% chance that any given neuron will be active.

- **All Neurons Active**
  - All neurons in the network are active and contribute to the computation of the final output.
- **Consistency with Training**
  - scaling ensures that the output of each neuron during inference is on average the same as its expected output during training, despite all neurons being active

Dropout During Inference

$$h_i' = h_i \cdot (1 - p)$$

Choosing Dropout Rates

Start with a Baseline

- A common starting point is to use a dropout rate of p=0.5 for fully connected layers.

Adjust for Different Layers

- For deeper layers or layers with more neurons, reduce the dropout rate to avoid excessive underfitting.
- Dropout rates between 0.2 and 0.5 are typical.
- For shallow layers or layers with fewer neurons, slightly higher dropout rates (up to 0.5) can be used.

Choosing Dropout Rates

Iterate

Experiment with different dropout rates and monitor the model's performance on a validation set.

Consider

Complex architectures (e.g., deeper networks, networks with many parameters) may require more regularization, so lower dropout rates might be appropriate.

Simpler architectures (e.g., shallow networks, networks with fewer parameters) may benefit from slightly higher dropout rates.

Avoid

excessively high dropout rates (e.g., p>0.5) as they can hinder the network's ability to learn effectively during training.

Practical Considerations

01

Layer-Specific Dropout

Adjust dropout rates based on the specific characteristics and depth of each layer in your neural network.

02

Experimentation

Iterate and experiment with different dropout rates to find the one that best suits your specific model architecture and dataset.

03

Validation Performance

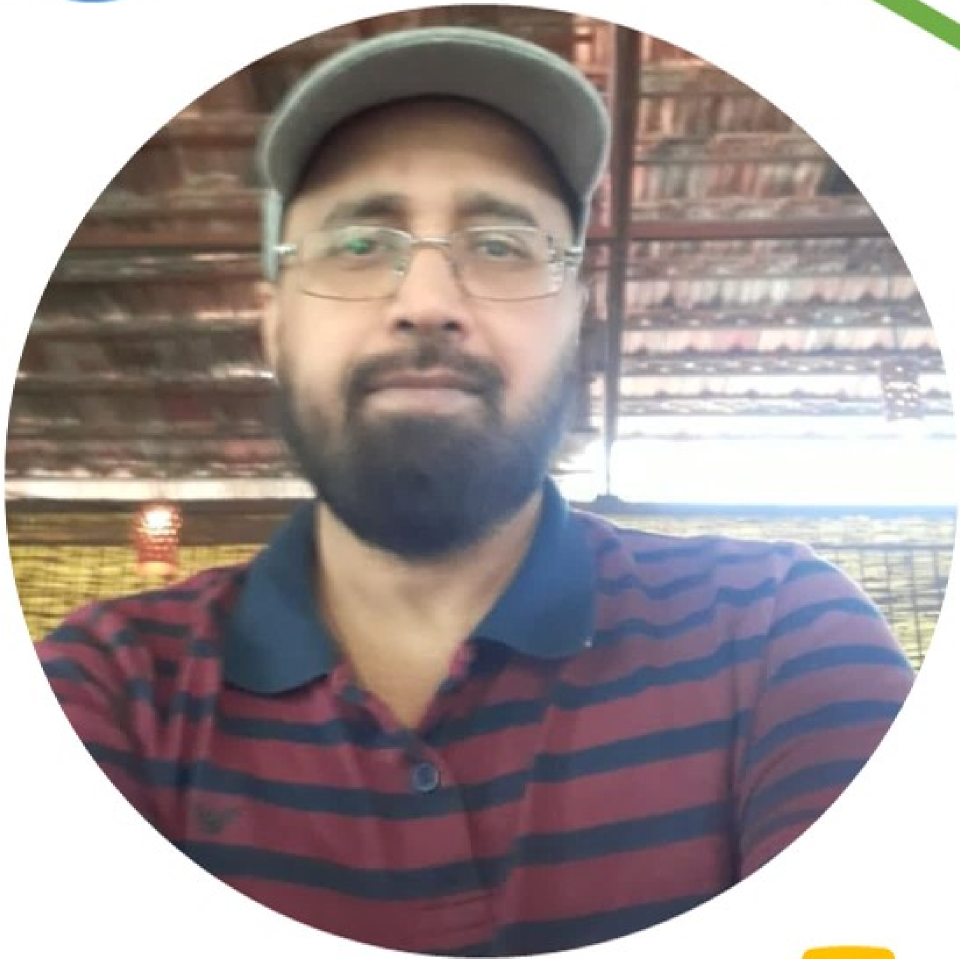
Always validate the chosen dropout rate on a separate validation set to ensure it improves generalization without sacrificing too much training accuracy.

Dropout vs L1 vs L2

Aspect	Dropout	L1 Regularization (Lasso)	L2 Regularization (Ridge)
Mechanism	Randomly sets a fraction ppp of the input units to zero during training.	Adds a penalty term	Adds a penalty term
Primary Effect	Reduces co-adaptation of neurons and improves generalization by training an ensemble of sub-networks.	Produces sparse models by driving less important weights to zero.	Prevents large weights, leading to smoother models.
Training Phase	Applies dropout and scales active neurons' outputs by 11−p\frac{1}{1 - p}1−p1.	Applies the L1 penalty to the loss function.	Applies the L2 penalty to the loss function.
Inference Phase	No dropout applied, uses the learned weights directly.	No change, uses the weights adjusted by L1 during training.	No change, uses the weights adjusted by L2 during training.

Dropout vs L1 vs L2

Aspect	Dropout	L1 Regularization (Lasso)	L2 Regularization (Ridge)
Hyperparameters	Dropout rate ppp.	Regularization parameter λ1	Regularization parameter λ2
Benefits	Improves robustness and generalization. Simple to implement.	Produces sparse models, useful for feature selection.	Prevents large weights, stabilizes training.
Challenges	Requires tuning of dropout rate, can slow down training.	Requires careful tuning of λ1, may not work well if all features are important.	Requires careful tuning of λ2, does not produce sparse models.
Applications	Widely used in deep learning tasks, especially in image and speech recognition.	Useful in scenarios where feature selection is important, such as linear models and sparse datasets.	Commonly used in regression tasks and deep learning to prevent overfitting.



Thanks !!

Happy Learning !