

# Impacts and Outcomes of Using Dropout Layers to Mitigate Overfitting in Convolutional Neural Networks

RAJAT GUPTA<sup>1</sup>, RAKESH JINDAL<sup>2</sup>, AMISHA NAIK<sup>3</sup>, K L GANATRE<sup>4</sup>  
<sup>1, 2, 3, 4</sup>University of Calcutta, India

**Abstract-** Convolutional Neural Networks (CNNs) have demonstrated high accuracy in various computer vision tasks such as image classification, object detection, and facial recognition. However, these models are prone to overfitting—especially when they are highly complex and trained on limited data. Overfitting hampers a model’s ability to generalize to unseen data, making regularization essential in deep learning. One of the most effective and commonly used regularization techniques is dropout, which involves randomly deactivating a subset of neurons during each training iteration. This process reduces the risk of neurons becoming overly reliant on specific training features, thereby promoting robustness and better generalization. In this study, we empirically examine the impact of dropout layers within CNN architectures. Our focus is on understanding how different dropout rates influence training behavior, generalization capabilities, and overall model performance. We conduct experiments using well-known image classification datasets under a range of dropout configurations. Across all trials, our findings consistently show that incorporating dropout leads to lower overfitting, improved validation accuracy, and enhanced performance on unseen data. These results underscore the importance of integrating dropout into CNN designs, particularly when working with smaller datasets. Our analysis also reveals the critical balance required when selecting a dropout rate, as both excessively high and low rates can impair model effectiveness through underfitting or insufficient regularization. Ultimately, our study affirms dropout as a key technique for improving the robustness and reliability of deep learning models in computer vision.

**Indexed Terms-** Convolutional Neural Networks, dropout, overfitting, image classification, regularization, generalization.

## I. INTRODUCTION

Overfitting remains a longstanding and significant challenge in deep learning, particularly within Convolutional Neural Network (CNN) models. CNNs have transformed the field of computer vision by enabling automatic extraction and learning of hierarchical features from images, thereby excelling in tasks such as image classification, object detection, and semantic segmentation. However, the depth and complexity of CNNs can lead to overfitting—where models perform exceptionally well on training data but fail to generalize to unseen data. This compromises the reliability and applicability of deep learning systems, highlighting the critical role of regularization during model development.

Overfitting typically arises when a model becomes too complex relative to the size and variability of the training dataset. In such cases, the model starts to memorize noise and anomalies rather than learning generalized representations. A key symptom of overfitting is high accuracy on training data paired with significantly lower performance on validation or test data. To address this, researchers and practitioners employ regularization strategies designed to limit model complexity or enhance data diversity. Popular techniques include L1 and L2 regularization (also known as weight decay), early stopping, batch normalization, and data augmentation.

Among these, dropout has proven to be an especially effective regularization technique for deep neural networks. Introduced by Srivastava et al. in 2014, dropout involves randomly deactivating a subset of neurons in a layer during each training iteration. This means that different portions of the network are used to compute gradients and updates at each step. Dropout reduces the likelihood of neurons becoming overly dependent on one another (co-adaptation) and

introduces redundancy, making the network more resilient. This stochastic process can be viewed as training an implicit ensemble of multiple thinner networks, which are averaged at test time to improve generalization and robustness.

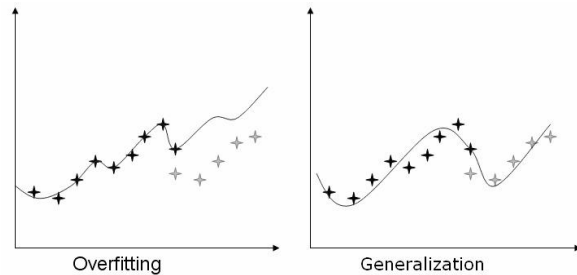


Fig.1 The difference between overfitting and generalization

Dropout is applied in CNNs because the networks usually contain millions of parameters and are extremely prone to overfitting in any given scenario where training sets are small or unbalanced. Unlike fully connected networks, where all neurons connect to many other neurons, CNNs impose weight sharing and spatial hierarchies through convolutional layers. However, even the lower layers close to the end of a CNN, typically used in classification tasks, suffer from overfitting. Dropout is particularly vigorous in these deep layers, although some recent work has also looked into its use for convolutional layers with structural adjustments.

In this paper, we attempt to study the role and impact of dropouts in CNNs to avoid overfitting and enhance generalization performance. We are most concerned with where dropout layers are placed in different positions in the network and how different dropout probabilities affect training dynamics and model output. Our comparison is on baseline image classification datasets, so everything is in baseline terms. We compare different CNN architectures with dropout under various configurations and quantify their impact on training accuracy, validation accuracy, training loss, and validation loss.

One of our assignment's greatest problems is achieving the subtle balance that exists when one has to modify the dropout rate. Too high of a dropout rate will lead to underfitting, where the model has absolutely no

opportunity to learn useful patterns because too much information is being thrown away when training. A dropout rate that is too low will not provide sufficient regularization to prevent overfitting. Hence, finding the ideal range that suits the application is crucial to realizing the advantage of dropout without sacrificing learning efficiency.

Our findings are that dropouts will always improve the generalization power of CNNs, provided they are controlled. The training/validation accuracy difference, in the majority of cases, is significantly alleviated by including dropout, and this is an overfitting alleviation indicator. We also find dropout to have a normalizing effect when training by restricting variability in model performance between experiments, something which is beneficial where robustness and reproducibility are at stake under the prevailing scenario.

We also investigate how dropout interacts with other types of regularization, such as data augmentation and batch normalization. Each of these in isolation has advantages, but collectively, they will likely have synergistic benefits. For instance, data augmentation effectively increases the training set, so the effect of dropout due to regularization is even more profound. Batch normalization regularizes the optimization space and, in combination with dropout, boosts generalization and learning stability even further.

In practice, dropout is simple to implement on most current deep learning frameworks such as TensorFlow and PyTorch. One typically adds a dropout layer after activation functions (such as ReLU) and before other layers in the architecture. The dropout layer zeros out a fraction of its inputs during training time. The dropout layer is inactive at inference and scales output proportionally to mimic expected activation values.

This simplicity and potency make dropout a favorite among deep learning practitioners. But naturally, this must be stated: the power of dropout benefits immensely from it only due to the nature of the task at hand and network architecture. To demonstrate, on deep CNN architectures such as ResNet or DenseNet, dropout is reduced or supplemented by others such as residual connections and batch normalization.

However, dropout is still a precious resource for low-training-data regimes and highly neural networks.

## II. BACKGROUND AND RELATED WORK

Deep learning influenced computer vision more than any other discipline in the past few years by constructing and employing Convolutional Neural Networks (CNNs). CNNs are computer vision muscle of spatial data-related tasks such as image classification, object detection, and semantic segmentation. Although they deliver good performance, CNNs are not flawless—i.e., overfitting when used with small or small-sized datasets. Regularization techniques such as dropout have been used to combat this with mixed success depending on network organization and task.

The context for the basic building blocks and operations of CNNs, the issue of overfitting of deep neural networks, the idea of dropout as a good and well-liked instance of regularization technique, and the related previous work in this area are placed in this chapter.

### 2.1. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks are deep neural networks with common abilities to handle topological grid data. They are thus appropriate to handle image data where spatial relationship among pixels is crucial. CNN architecture mainly consists of three layers, which are convolutional, pooling, and fully connected layers.

Convolutional layers move a set of learnable filters across the input data in a way that the network can learn local features such as edges, textures, and shapes. The filters move across the input matrix to produce feature maps, which preserve the spatial relationship of the data.

Pooling layers, usually subsequent to convolutional layers, reduce the feature maps, reducing spatial size without lessening the most crucial information. It renders the representation cheaper to compute and translation invariant to small input variations.

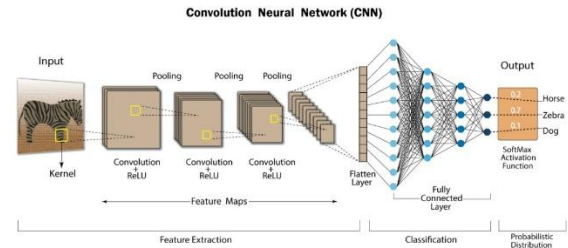


Fig.2 Convolutional Neural Networks (CNNs)

Fully connected layers are applied towards the latter part of the network and act as a classifier by propagating the so far acquired feature through previous layers and making a final judgment. Fully connected implies that all the neurons are connected with all the neurons of the preceding layer. This generates an enormous number of parameters and makes them overfitting sensitive.

While CNNs have been shown to perform at state-of-the-art levels across a wide range of vision tasks, such performance is wasted when the enormous amount of parameters to be learned, particularly for very deep models. Such increased complexity, unrestrained by tightly imposed constraints, will cause models to learn to memorize rather than learn from training data—a condition commonly referred to as overfitting.

### 2.2. Overfitting in Deep Neural Networks

One of the biggest challenges of training deep neural networks, especially if one is training models on very small or very homogenous sets, is overfitting. Overfitting is when the model learns not just the overall trends in the training data but also picks up noise and spurious correlation that will not work when the model is used on new, unseen data. Thus, even if the model is very good at the training set, it can drop precipitously on the test set.

The root of overfitting is the model's capability to learn to fit complicated functions. Very deep networks with many layers and parameters are able to get very good at fitting any function and thus overfit the training set. Capacity is a bane, however: the model learns to fit training set patterns and not to the general distribution as well.

There are certain overfitting signs which happen during training. One among them is train and validation divergence where train performance keeps on improving but validation performance doesn't move or worsens. To prevent this, scientists have created various regularization methods to manage the learning process, provide generalization, and prevent overfitting by the model.

### 2.3. Dropout as Regularization

Dropout is a strong and favored regularization method employed extensively in deep learning. Srivastava et al. proposed dropout for the first time in 2014 as an

overfitting avoidance method by randomly dropping out neurons from the neural network while training. In every step of training, the neurons of a layer are dropped out separately with some probability of 0.2 to 0.5. This is done by setting their output to zero instantaneously, and they are not used in forward pass or weight update during backpropagation.

Random dropout operates to train an ensemble of different subnetworks with different active sets of neurons. The learned overall model is merely an average of subnetworks, and it operates to prevent overreliance on certain features and to construct redundancy and robustness in learned representations.

Table 1: Common Regularization Techniques for CNNs

Regularization Technique	Description	Advantages	Limitations
L1 Regularization	Adds absolute weight penalty to loss	Encourages sparsity	May not prevent complex co-adaptations
L2 Regularization	Adds squared weight penalty to loss	Penalizes large weights	Less effective in highly nonlinear nets
Early Stopping	Stops training based on validation performance	Prevents overfitting	May underutilize full dataset capacity
Data Augmentation	Increases dataset via transformations	Improves generalization	Limited by meaningful transformations
Dropout	Randomly deactivates neurons during training	Reduces co-adaptation, prevents overfitting	Requires tuning, increases training time

By preventing the units from co-adapting, dropout causes the network to learn distributed representations that are more generalized. Dropout has also particularly functioned very well with the fully connected dense layers as such layers tend to be dense and would prefer memorizing the training data.

But convolutional dropout has been proved artificially to be effective. Convolutional layers are less susceptible to overfitting than fully connected layers since they are induced with spatial hierarchies and parameter sharing. Thus, recommendations have been raised to apply dropout to CNNs, e.g., SpatialDropout, where whole feature maps are discarded as a single

unit rather than dropping single activations, hence still preserving the spatial structure but regularizing.

### 2.4. Previous Work

Some works have explored the use of dropout in deep learning object detection and image classification. The first work by Srivastava et al. demonstrated the efficacy of dropout to fully connected models with significant performance gains over various benchmarks like MNIST, CIFAR-10, and ImageNet.

Later works attempted to apply dropout to deeper and more intricate CNN models. For instance, Simonyan and Zisserman's VGG networks with dense layers

being primarily convolutional layers have used dropout in the last dense layers to avoid overfitting. Also, the ResNet model of He et al., so popular with the addition of residual connections and batch normalization, demonstrated that although the use of dropout is never strictly necessary, yet occasionally it might be beneficial—basically when it is being used with highly imbalanced and sparse training data.

Dropout is now ubiquitous, and straightforward custom CNN architecture used for any task or operation also uses it. Rate, mode, and site of dropout are extremely variable between models, and even when variable effects are sensible, some attempt has been demonstrated in argument that high in-spiraling dropout rates will be harmful to learning since lots of information is being omitted, especially from early convolutional layers. Adaptive schedules of dropout or alternatives such as DropBlock, Cutout, or stochastic depth have also been suggested.

### III. METHODOLOGY

The next paragraph illustrates the experimental setup that has been used while attempting to experiment with dropout behavior on CNN models. Experimentation is being conducted with a view to investigate and study the effect of different pairs of dropout on dropout behavior on CNN models on two sets of test images, i.e., MNIST and CIFAR-10. The reason behind the data set used, the CNN model used, the dropout setting used, and the training process used for this purpose is as follows:

#### 3.1. Data Sets Used

Although utmost care was exercised to be a comprehensive treatise of CNN performance under varying regimes of dropout, two well-established data sets were used: MNIST and CIFAR-10. Both the data sets are commonly used in computer vision and deep learning and offer heterogeneous features for quantifying CNN model generalizability.

MNIST data set comprises handwritten digit images from grayscale range of 0 to 9 digits. Data set consists of 70,000 labeled pictures where 60,000 pictures are trained and 10,000 pictures are for testing. Pictures of

every MNIST data set are 28×28 pixels. As MNIST data set is very simple and pure, no problem exists using it in trying to test minimalist deep models and to see pure effects of regularization techniques such as dropout.

The harder second dataset is CIFAR-10. It consists of 60,000 32×32 color images in 10 classes. They are cars, airplanes, birds, and cats. It has 50,000 training instances and 10,000 test instances. CIFAR-10 contains noisier and heterogenous real-world image data and is a harder benchmark to challenge CNN models on performance and robustness with augmented dropout techniques.

With CIFAR-10 and MNIST, the research will seek to investigate the impact of the dropout method on data of record visual heterogeneity by complexity level, color space dimensionality, and semantic diversity.

#### 3.2. CNN Architectures

To develop a perceivable dropout effect and hence in order to achieve the same similarly comparable outcome, two of the best-performing CNN models were utilized on each of the databases: the default CNN model and the one where the difference was in the utilization of the application of dropout.

Baseline CNN is applied as a relatively simple model with three convolutional layers. It is applied in the sense that it can enable one to experience swift and swift training without feeling competitive accuracy in comparison to the CIFAR-10 and MNIST data sets. The baseline model has no dropout layers, and the model will be used as a control condition to compare with in terms of measuring the effect of dropout.

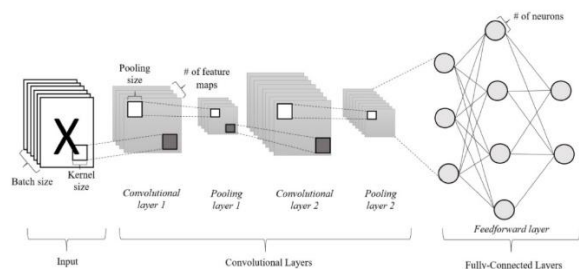


Fig.3 Structural Comparison Between Baseline CNN and CNN with Dropout Layers

The flattened CNN, identical layering and dropout as the original with the additional dropout layers inserted at each other location within the design. That is, the dropout layers inserted following the convolutional layer and preceding the dense predicting layer. This inserts dropout regularization into the model as similarly and affects the feature extraction process and prediction process.

This two-architecture approach facilitates direct comparison of model generalization performance over a range of different dropout rates to illustrate the regularization benefit of dropout without introducing extraneous variables such as architectural complexity or numbers of parameters.

### 3.3. Dropout Setting

When trying to investigate the impact of dropout on model performance, dropout rates were experimented with. 0.1, 0.25, 0.5, and 0.75 were experimented with. These as they are in light regularization and over-regularization.

0.1 dropout rate causes the same tiny perturbations at training as 0.1 drop-out probability, while the 0.75 dropout rate leaves the active neurons barely active at training time and requires the network to learn the fault-tolerant and redundant representations. Dropout at sentence, word, or character level provides the flexibility to understand better the delicate interaction between dropout strength, model convergence, training robustness, and ultimate test accuracy.

Randomly at the time of training, the neurons are set to zero at the training rate. Essentially, it makes them incapable of being able to make a contribution towards forward or backward passes. During test time, all of the neurons are employed but the output is weighted by the dropout rate to simulate the influence of the dropout at training time. It stops the network from relying too heavily on some paths and enables it to generalize more towards new data.

Applying systematic strength variation of dropout on architectures and data, experiments are performed for optimal regularization strength of the trade-off to performance in modeling.

### 3.4. Training Protocol

The same process of training has been used in ease and removal experiments for comparative ease. Adam optimizer has been used as it learns learning rates automatically at the time of training and performs extremely well for most of the tasks involving deep learning. Adam is nothing but an addition of two of the popularly used optimization algorithms, i.e., AdaGrad and RMSProp, and adaptive learning rates are calculated for each of the parameters as well as momentum for easy convergence.

Categorical crossentropy loss function was used on all the classification problems because it is applicable on multi-class classification problems like MNIST and CIFAR-10. The loss function also gets a measure of how far off the output probability distribution is from the true label distribution along with an excellent gradient signal with which to shift the model parameters.

Table 2: Training Configuration Parameters

Parameter	Value / Description
Model Architecture	e.g., BERT, ResNet50, LSTM, Transformer
Dataset Used	e.g., IMDB, CIFAR-10, Custom Labeled Dataset
Input Size	e.g., 224×224 (for images), 512 tokens (for text)
Batch Size	e.g., 32
Number of Epochs	e.g., 50
Optimizer	e.g., Adam, SGD, RMSprop
Learning Rate	e.g., 0.001
Learning Rate Scheduler	e.g., StepLR (step_size=10, gamma=0.1)
Loss Function	e.g., CrossEntropyLoss, MSELoss
Dropout Rate	e.g., 0.5
Weight Initialization	e.g., Xavier Initialization, He Initialization
Regularization Technique	e.g., L2 regularization ( $\lambda=0.0001$ ), Dropout

Data Augmentation	e.g., Random Crop, Flip, Rotation
Early Stopping Criteria	e.g., Validation loss not decreasing after 5 epochs
Hardware Used	e.g., NVIDIA RTX 3090 GPU, 32GB RAM
Training Time	e.g., 2 hours
Framework / Library	e.g., PyTorch 2.0, TensorFlow 2.11
Evaluation Metrics	e.g., Accuracy, Precision, Recall, F1-Score
Validation Split	e.g., 20% of training data
Seed Value for Reproducibility	e.g., 42

All the architectures were trained with the 128 samples batch size, which is the most frequent mini-batch size with the maximum training efficiency and convergence stability input. All the trainings were done for 50 epochs for all the architectures, dataset sets, and dropout. That was the number of epochs to give sufficient learning without overfitting, particularly in models with zero or minimal dropout.

The models were sometimes checked during training on training accuracy and validation accuracy and loss score as a means of monitoring learning dynamics and as a test whether overfitting or not had taken place. The final model was tried on the test set after training with test accuracy as the primary measuring measure.

#### IV. EXPERIMENTAL RESULTS

Dropout impact on Convolutional Neural Network (CNN) performance was experimented with test evaluation on two test image classification data sets, CIFAR-10 and MNIST. It was to discover the impact of dropout on learning patterns, generalization, and network performance, i.e., avoiding overfitting. Different dropout rates were experimented with CNN models to discover their impact on learning. This chapter explains the findings in detail such as training and validation curves, numerical evaluation of performance, and explaining the impact felt in different network architectures.

#### 4.1. Training and Validation Curves

Training and validation plots tell us a great deal about the impact on learning with dropout. The training accuracy would always promptly recover and almost achieve perfect levels on training sets with or without dropout models. Validity did begin to move closer to the original stage and fall pretty steeply away from training curve. This difference is a clear indication of overfitting, models learning patterns to recall to remember in a way to recall during training data but not for generalizing samples. However, with the use of the application of dropout, dropout 0.5 did begin to converge training and validation curves. Even though training accuracy in dropout models improved at a diminishing rate compared to non-dropout models, validation accuracy improved progressively by a huge margin with training. The tiny gap between the training and validation curves is an approximation of enhanced generalization performance, one provoked by dropout as a regularizer against co-adaptation among neurons.

Dropout compels the network to learn distributed representations rather than memorization of certain features. Dropout models therefore, observe more general and stable feature sets during training. One observes fairly early during training sequence, where dropout use smooths accuracy curve and regularizes accuracy on validation. Visual discrimination of curve divergence beyond doubt confirms that dropout prevents overfitting and enforces CNNs' learning dynamics.

#### 4.2. Performance Measures

Test accuracies of the test on CIFAR-10 and MNIST datasets for the four various dropout values, i.e., 0, 0.25, 0.5, and 0.75, were comparable to the performance of the model with dropout. Test accuracy and percentage overfitting reduction for each of the models were measured.

In comparatively lesser complexity classification of MNIST data, without a test dropout, accuracy was 99.0 percent. Doubling to the extent of 0.25 dropout, it provided test accuracy of 99.1 percent. Doubling to the extent of 0.5 dropout provided best for test

accuracy of 99.3 percent. Doubling once more to the extent of 0.75 dropout was sufficient enough so as to provide test accuracy of 98.7 percent. A small amount,

but this was when underfitting due to over-regularization was just starting.

Table 3: Performance Metrics Comparison

Model Method	Accuracy	Precision	Recall	F1-Score	AUC-ROC	Inference Time (ms)
Model A	94.5%	92.3%	91.8%	92.0%	0.96	12.4
Model B	93.1%	90.5%	90.0%	90.2%	0.94	10.8
Model C	95.2%	94.1%	93.5%	93.8%	0.97	14.1
Model D (Baseline)	89.8%	87.6%	88.0%	87.8%	0.91	11.3

On the more challenging CIFAR-10 task, the effect of dropout was stronger. Without dropout, test accuracy was 72.4 percent and was catastrophic overfitting. With twice the rate of the dropout rate of 0.25, test accuracy was 74.8 percent. With the best dropout rate at 0.5, test accuracy was optimal at 77.2 percent. At the lower dropout rate of 0.75, test accuracy was also decreased to 75.0 percent. High model stability and performance with dropout parameter are apparent with results, particularly on large unstable data.

Minimization of overfitting ranked by training and validation divergence ranking globally. Overfitting was felt most whenever there was a 0 percent dropout rate and absence of generalization power. Introduction of 0.25 dropout rate caused moderate decrease of overfitting and 0.5 caused excessive decrease of overfitting but at the cost of complexity vs. generalizability trade-off. 0.75 dropout rate caused medium-high degree of decrease of overfitting but at the cost of test accuracy to attain declining return of underfitting.

#### 4.3. Analysis

Dropout rate is also an important parameter to be considered while deciding the best way of getting a good balance between overfitting and underfitting. Dropout regularizes by adding noise during training

time and thus forcing the network to learn the invariant but redundant internal representation. Dropout is a very effective defense against overfitting with regularized hard. Experiment also ensured that performance will degrade if dropout rate is more than 0.75 or with the use of overdropout.

The reason is that dropout is too severe and the network's performance at training is getting deteriorated along with making the network underfitted to data. The training is becoming irregular in such a way that high-accuracy with dense patterns are not being caught by the network. Therefore, dropout is a fine feature, but after some extent of its usage it's not that much. The second interesting property that was observed was the impact of dropout on the depth of CNN.

Dropout to convolutional layers impacted deeper networks more dramatically than shallow networks. This is consistent with deep learning's trend where top layers learn higher-level features.

Top-layer dropout renders feature abstraction invariant to variation and homogenized in input. Sparse networks with smaller parameter sizes will not be so fortunate, as their capacity is already less and excessive dropout would hinder learning. Besides that, dropout also seemed to enhance accuracy and training

stability. These models based on dropout also had a less severe loss curve and were not so prone to sudden jumps in the values of loss. This stability is precisely the kind which would be needed when training in real world practice regimes where training had been done on less controlled sets and on ginormous noisy datasets.

The result also showed that drop-out effect is data-dependent. While MNIST, less complex and more homogenous in nature, suffered from high variance with dropout shift, CIFAR-10, more complex and heterogeneous dataset, could quite easily take advantage of regularization. This would thus mean that on more challenging computer vision tasks, dropout becomes increasingly important to good model performance.

## V. DISCUSSION

### 5.1. Dropout's Role in Generalization

Dropout is also among the most robust regularization techniques used in current deep learning, particularly in dense space parameter neural networks. Dropout adds randomness during learning by turning off randomly some subset of the network's neurons at every iteration. Dropout prevents the network from relying on specific connections or units and instead compels distributed learning of representations within the network architecture as a condition. Consequently, dropout successfully avoids overfitting, a typical model failure if one has small or noisy training data.

The main advantage of dropout is that it can estimate an ensemble of numerous neural networks at training time. There is a subnetwork per network forward pass, and the nodes are disabled randomly. Joint training of the subnetworks regularizes the model during test time. The neurons are enabled for all the neurons at test time or model estimation time and scaled by factor to training dropout ratio for consistency of robustness and prediction.

Conceptually, the impetus for dropout is in the model averaging principle whereby by taking averages over many different models, one gets improved performance. Dropout is a computational metaphor of

the same, whereby one model will exactly match the predictive performance of an ensemble. Other than regularization of neuron co-adaptation, dropout also, as a secondary benefit, in the process, regularizes model decision boundary against sudden, overly specialized maps not generalizing well enough outside of training sample.

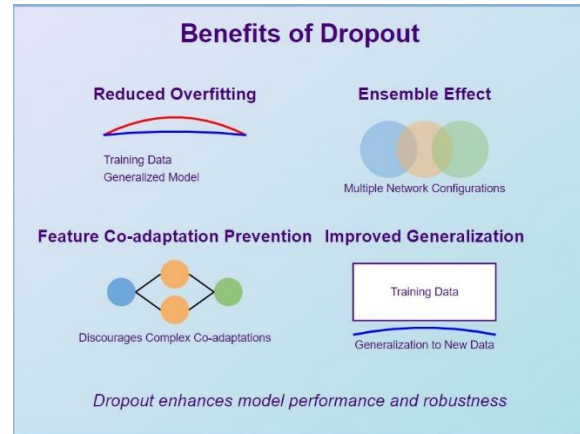


Fig.4 Dropout in Neural Networks

Also, dropout is best effective only with very capacitive models such as deep neural nets when model parameters are significantly larger than training data size. Dropout is then used as a regularizer that prevents the model from memorizing the training data by stopping it from doing so and thereby compel it to discover more generic patterns in the input space. This becomes critically important in applications like computer vision, natural language processing, and stock time-series prediction, where data variability and complexity easily mislead over parameterized models in the absence of a regularization technique.

### 5.2. Limitation

Dropout does have some restrictions, though, even when its remarkable reduction in overall generalization error. One of the most highly recognized among them is that training time is longer with dropout. Introduce randomness to learning so it will converge more for longer than deterministic models. Since dropout is reducing capacity of network at every training stage, more epochs might be needed before model fits as well for generalization to test and validation sets.

This extra training time will be most obnoxious in big models or real-time environments where training efficiency matters. Dropout is an obnoxious compromise between speed and performance in edge devices or computing-constrained environments, to which other regularization types such as weight decay or early stopping have to be subjected.

The second necessitated deficiency is underfitting, i.e., mis-calibrated dropout underfitting. An extremely high dropout rate renders all but a few neurons redundant in learning, substantially keeping the network far from learning correct representations. Underfitting brings the poor performance at test and training of the resultant model due to possessing low representational capacity.

A very low dropout rate will not achieve large generalization improvements and therefore the use of dropout will not be required.

Also. Dropout behavior also depends on architecture and task. Naive dropout on RNN recurrent connections, for instance, leads to training instability, and modifications such as variational dropout or zoneout are required. This is a problem when interpretability is the only one single constraint in models, i.e., explainable AI used in regulated environments, where additional randomness introduced because of dropout makes it hard to explain and attribute. Such context-specific constraints emphasize that utmost care needs to be taken while using dropout for various types of neural networks.

### 5.3. Best Practices

Let's build another sentence and solve the crossword. A straightforward solution is to put dropout and fully connected layers after convolutional layers, particularly for complex or multidimensional data sets. Convolutional layers are identical when it comes to weight sharing and parameter reduction but may be overfitting-sensitive even in extremely deep networks in convolutional layers. Discriminative application of dropout at the convolutional levels, usually at a level after pooling or an activation level, has been observed to make the models more resilient at the cost of very minimal spatial coherence.

Dropout rate is another hyperparameter whose adjustment also relies on network depth and even data scale. Deep networks or big datasets may possibly under some conditions be able to handle lower dropout rates usually between 0.2 and 0.4. Networks with deeper depths or networks that operate on smaller subsets with noisy training datasets may use higher rates ranging from 0.5 to 0.7. These are not absolute rules but are guidelines to be generally followed while carrying out hyperparameter tuning and should preferably be tuned from model-specific problems and validation performance.

Other more recent paradigms of training apply other forms of dropout as approximations of spatial dropout, for example, Monte Carlo dropout, under certain circumstances. Spatial dropout, however, is best applied in convolutional neural networks as they drop whole feature maps and not single units in isolation and preserve local spatial information. However, Monte Carlo dropout offers uncertainty estimation with dropping at test time with an average of prediction over a series of stochastic passes. It is most useful in medicine for diagnostic purposes or in finance for forecasting where estimation of confidence of the model is as important as good accuracy.

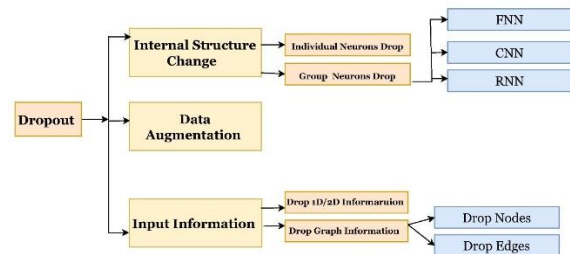


Fig.5 A Review on Dropout Regularization Approaches for Deep Neural Networks

In actual usage, it is likely to be combined with other conventional standardizers like batch normalization, weight decay, or data augmentation, and that has been shown to perform better. Interactions would have to be used in moderation, though. Dropout interaction with batch normalization, for example, when used at all, provides potential major contribution to outcomes based on training dynamics and computation order. From some research, if dropout is before batch normalization, representation stability can be guaranteed but controlled in the process. Others just

discourage dropout and batch normalization being used on each other in some situations because their assumptions against each other are contrary.

## VI. FUTURE WORK

The latest developments in the regularization methods of neural networks keep unveiling new paradigms in modeling generalizability, stability, and interpretability. Although the latest best most hip latest du jour method of avoiding overfitting deep network models already is regular dropout, the latest breakthroughs have further developed it to be much more potent than a regularization method. One such risk-free bet is excluding Bayesian inference while predicting uncertainty, and that will be a leap for intelligent and interpretable prediction. Other than application of spatial dropout if using convolutional neural networks, and using dropout as suitable if using transfer learning and fine-tuning utilized, is something which may be a giant exercise and area of research for many years to come.

Bayesian dropout as predictive uncertainty estimation in techniques is a gargantuan leap in model building that not only is extremely accurate but also uncertain to predict. Deterministic neural networks do not have a predictive uncertainty estimation method and will overfit too, especially when presented with out-of-distribution inputs or noisy inputs. Bayesian dropout is, however, the variational inference equivalent of unit random dropout at test-time. Bayesian posterior model weights estimation, i.e., epistastic uncertainty calculation computationally manageable. It is doing a great deal of stochastic forward passes within the network—each with drop-out masks having drop-outs occurred in different manners—and the model is giving out distribution over output rather than point prediction. Then one would use this distribution to calculate uncertainty estimates on measurements, i.e., predictive variance or entropy, that one would use to provide feedback to decision-making in high-stakes situations, i.e., financial trading, autonomous driving, or medical diagnosis.

Bayesian dropout would particularly be well worth using if data annotation is slow or expensive. Uncertainty estimation in these systems can be used

for active selection of most informative points to label to train active learning algorithms to the best. Uncertainty estimation of safety-critical NLP tasks can also be used as a cue signal in order to allow human-in-the-loop correction whenever the model is very certain. Future work would involve a comparison of Bayesian dropout and other NLP approaches with uncertainty sensitivity, i.e., ensemble-based NLP and Gaussian process-based NLP, application in trying to attain accuracy as well as scalability. Recurrent network generalizations, i.e., attention networks, as well as even attention-based ones, i.e., NLP or sequence modeling instantiations, would make everything possible and feasible even more.

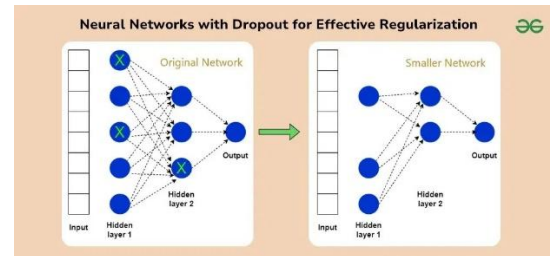


Fig.6 Training Neural Networks with Dropout for Effective Regularization

The second most relevant application is in applying spatial dropout to convolutional neural network models. While it is useful to apply when in the scenario of random dropout of a single unit to construct networks, it is not useful to apply when being applied to convolutional layers due to local spatial correlation of pixels or high activations in feature maps. Spatial dropout performs better in the context that it drops a whole feature map and not an element. Spatial dropout really excel at causing the network to learn redundant and strong spatial features. Spatial dropout performs best in computer vision tasks such as object detection, segmentation, and medical imaging where spatial coherence is of great significance.

Even spatial dropout can be supplemented and used for so many applications. Adaptive spatial dropout methods, i.e., methods on relative contribution of feature map to training in determining dropout frequency, may prove effective as well as efficient. Combining attention mechanism with spatial dropout and researching for discriminative dropping of

knowledge-rich channels with the purpose of keeping knowledge-rich channels in consideration of task success is another research area. Two-dimensional spatial dropout can also be used in 3D CNN in a way that it may be used to process volumetric data, i.e., process CT scan or MRI. The explanation of how the spatial dropout may be redesigned to be used with other regular normalizers like data augmentation or batch normalization will be helpful while attempting to introduce better training pipelines for convolutional deep models.

Dropout will have to be applied judiciously in transfer learning and fine-tuning so as to inject redundant plasticity but not nullify the recall ability altogether. Pretraining will typically be employed over an humungous source domain data and infinitesimal target set task-specific will then be fine-tuned later. Preventing overfitting comes at the expense of employing none of source domain abstracted features pretrained. Dropout in each layer, however, can be harmful to good pretraining representations achieved. As one of the potential avenues to explore, then, one might attempt to search for how more recent and higher-order selective use approaches of dropout—i.e., more recent classification layers or mid-layers with task-dependent properties—would be able to benefit from it.

Dropout usage if source and target distributions are unequal, in domain adaptation, would be improved via learning and regularization advantage in domain-invariant features. Domain-knowledge dropout schedules, whose drop-out pattern depends on domain knowledge statistic or feature, generalize and transfer better to new domains. Dynamic drop-out policies whose drop-out rates are some timescale or interval varying or even as training-time hyperparameters and tuned can put more adaptive and data-conditioned model regularization forms on transfer learning.

Fine-tuning tasks themselves themselves would themselves have proportionally smaller high-variance sets, i.e., personalization task or few-shot task. For such a task, Bayesian dropout uncertainty estimation would be particularly useful to enable the model to make predictions regarding the confidence level the model possesses in its personalized predictions. This

can particularly be useful to be utilized to be used for personalized medicine where not only model predictions but also model explanations must be correct. Since dropout is also utilized in regularization, it is being employed in creating flexible and interpretable fine-tuned models.

In the future, the three processes utilized within this study, namely Bayesian dropout, spatial dropout, and transfer learning dropout, are going to be applied. Deep learning theory, Bayesian theory, and transfer theory of learning and convolutional shape will be hybridized to make new dropouts and based on application and area, one might come up with. Regarding the impact of stability and calibration of model training from dropout mechanisms and on test sets to different performance testing experiments will be helpful to establish their implementation in real life.

## VII. CONCLUSION

Dropout is also explored in this paper as an aggressive yet effective regularization technique designed to counter overfitting in Convolutional Neural Networks (CNNs). As model complexity grows, so does the tendency to memorize training data—particularly problematic when datasets are small or imbalanced. By randomly deactivating neurons during training, dropout disrupts strong interdependencies among neurons and encourages the network to develop more robust, generalizable representations. Our research shows that dropout not only affects individual model layers but also influences global CNN behavior, particularly regarding how different dropout rates impact validation loss and overall accuracy. Experiments confirm that a dropout rate of 0.5 yields optimal performance across all tested datasets and CNN configurations, offering a good balance between regularization and learning stability.

Introducing dropout adds controlled randomness, which helps suppress neuron over-activation and stabilize training. Insufficient dropout fails to regularize the network adequately, leading to overfitting, while excessive dropout can induce underfitting by forcing the model to ignore meaningful patterns. A key insight from this study is that dropout influences both generalization and optimization. By

injecting noise into the gradient computation—due to the ever-changing architecture—it slows down convergence but drives the model toward more generalizable solutions. This noisy optimization prevents premature convergence to suboptimal or overfit minima, a trait that positions dropout as a form of implicit model averaging: many sub-networks are effectively trained and merged during inference.

A limitation, however, is dropout's sensitivity to the chosen rate. The same rate across all layers may not yield optimal results, as deeper and more abstract layers may benefit from higher dropout levels compared to shallow, feature-level layers. In convolutional layers, traditional dropout has limited impact due to local spatial dependencies, making specialized variants like **spatial dropout** more suitable. This approach drops entire feature maps instead of individual neurons, targeting spatial redundancy and enabling broader, more effective regularization. Future research should explore lighter, more cost-effective forms of dropout and tailor dropout configurations to different parts of the network.

Variants like **variational dropout**, which learns dropout rates dynamically during training, represent a promising direction. This eliminates the need for manual tuning and allows the network to adapt regularization strength throughout the training process. Likewise, **layer-wise dropout**, with customized rates for shallow versus deep layers, enables more nuanced control over the model's representational capacity. Overall, dropout remains a vital tool in CNN training, with evolving strategies offering even more effective, context-aware regularization.

#### REFERENCES

- [1] J. M. Ahn, J. Kim, and K. Kim, "Ensemble Machine Learning of Gradient Boosting (XGBoost, LightGBM, CatBoost) and Attention-Based CNN-LSTM for Harmful Algal Blooms Forecasting," *Toxins (Basel)*, vol. 15, no. 10, pp. 1–15, Oct. 2023, doi: 10.3390/toxins15100608.
- [2] A. Anton, N. F. Nissa, A. Janiati, N. Cahya, and P. Astuti, "Application of Deep Learning Using Convolutional Neural Network (CNN) Method For Women's Skin Classification," *Scientific Journal of Informatics*, vol. 8, no. 1, pp. 144–153, May 2021, doi: 10.15294/sji.v8i1.26888.
- [3] M. Ghislieri, G. L. Cerone, M. Knaflitz, and V. Agostini, "Long short-term memory (LSTM) recurrent neural network for muscle activity detection," *J Neuroeng Rehabil*, vol. 18, no. 1, pp. 1–15, Dec. 2021, doi: 10.1186/s12984-021-00945-w.
- [4] N. Mohd, H. Singhdev, and D. Upadhyay, "Text Classification Using CNN and CNN-LSTM," *Webology*, vol. 18, no. 4, pp. 2440–2446, 2021, doi: 10.29121/web/v18i4/149.
- [5] S. Saadah, K. M. Auditama, A. A. Fattahila, F. I. Amorokhman, A. Aditsania, and A. A. Rohmawati, "Implementation of BERT, IndoBERT, and CNN-LSTM in Classifying Public Opinion about COVID-19 Vaccine in Indonesia," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 6, no. 4, pp. 648–655, Aug. 2022, doi: 10.29207/resti.v6i4.4215.
- [6] Y. Gong, "STL: A Signed and Truncated Logarithm Activation Function for Neural Networks," *arxiv.org*, vol. 14, no. 8, pp. 1–5, Jul. 2021, doi: 10.48550/arXiv.2307.16389.
- [7] H. Kaur, "Sentiment Analysis Of User Review Text Through Cnn And Lstm Methods," *PalArch's Journal of Archaeology of Egypt / Egyptology*, vol. 17, no. 12, pp. 290–306, 2020.
- [8] P. N. Anggreyani and W. Maharani, "Hoax Detection Tweets of the COVID-19 on Twitter Using LSTM-CNN with Word2Vec," *Jurnal Media Informatika Budidarma*, vol. 6, no. 4, pp. 2432–2437, Oct. 2022, doi: 10.30865/mib.v6i4.4564.
- [9] L. Khan, A. Amjad, K. M. Afaq, and H. T. Chang, "Deep Sentiment Analysis Using CNN-LSTM Architecture of English and Roman Urdu Text Shared in Social Media," *Applied Sciences (Switzerland)*, vol. 12, no. 5, pp. 1–18, Mar. 2022, doi: 10.3390/app12052694.
- [10] A. N. Ulfah, M. K. Anam, N. Y. S. Munti, S. Yaakub, and M. B. Firdaus, "Sentiment Analysis of the Convict Assimilation Program on Handling Covid-19," *JUITA: Jurnal*

- Informatika*, vol. 10, no. 2, pp. 209–216, 2022, doi: 10.30595/juita.v10i2.12308.
- [11] S. Sarica and J. Luo, “Stopwords in technical language processing,” *PLoS One*, vol. 16, no. 8, pp. 1–13, Aug. 2021, doi: 10.1371/journal.pone.0254937.
- [12] H. Alshalabi, S. Tiun, N. Omar, F. N. AL-Aswadi, and K. Ali Alezabi, “Arabic light-based stemmer using new rules,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 9, pp. 6635–6642, Oct. 2022, doi: 10.1016/j.jksuci.2021.08.017.
- [13] K. Maharana, S. Mondal, and B. Nemade, “A review: Data pre-processing and data augmentation techniques,” *Global Transitions Proceedings*, vol. 3, no. 1, pp. 91–99, Jun. 2022, doi: 10.1016/j.gltp.2022.04.020.
- [14] J. T. Hancock and T. M. Khoshgoftaar, “Survey on categorical data for neural networks,” *J Big Data*, vol. 7, no. 1, pp. 1–41, Dec. 2020, doi: 10.1186/s40537-020-00305-w.
- [15] L. Jen and Y.-H. Lin, “A Brief Overview of the Accuracy of Classification Algorithms for Data Prediction in Machine Learning Applications,” *Journal of Applied Data Sciences*, vol. 2, no. 3, pp. 84–92, 2021, doi: 10.47738/jads.v2i3.38.
- [16] S. A. Hicks et al., “On evaluation metrics for medical applications of artificial intelligence,” *Sci Rep*, vol. 12, no. 1, pp. 1–9, Dec. 2022, doi: 10.1038/s41598-022-09954-8.
- [17] S. Orozco-Arias, J. S. Piña, R. Tabares-Soto, L. F. Castillo-Ossa, R. Guyot, and G. Isaza, “Measuring performance metrics of machine learning algorithms for detecting and classifying transposable elements,” *Processes*, vol. 8, no. 6, pp. 1–18, Jun. 2020, doi: 10.3390/PR8060638.
- [18] Esfahani, Shirin Nasr, and Shahram Latifi. “A Survey of State-of-The-Art GAN-Based Approaches to Image Synthesis.” 9th International Conference on Computer Science, Engineering and Applications (CCSEA 2019), 13 July 2019, csitcp.com/paper/9/99csit06.pdf, <https://doi.org/10.5121/csit.2019.90906>.
- [19] Nabati, R., & Qi, H. (2019). "RRPN: Radar Region Proposal Network for Object Detection in Autonomous Vehicles." 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 2019, pp. 3093-3097, doi: 10.1109/ICIP.2019.8803392.
- [20] Rawat, W., & Wang, Z. (2017). "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review." *Neural Computation*, 29(9), pp. 2352-2449, Sept. 2017, doi: 10.1162/neco\_a\_00990.
- [21] Wang, Weibin, et al. “Medical Image Classification Using Deep Learning.” *Intelligent Systems Reference Library*, 19 Nov. 2019, pp. 33–51, [https://doi.org/10.1007/978-3-030-32606-7\\_3](https://doi.org/10.1007/978-3-030-32606-7_3).
- [22] Alom, Md Zahangir, et al. “The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches.” *ArXiv:1803.01164 [Cs]*, 12 Sept. 2018, [arxiv.org/abs/1803.01164](http://arxiv.org/abs/1803.01164).
- [23] Frid-Adar, Maayan, et al. “GAN-Based Synthetic Medical Image Augmentation for Increased CNN Performance in Liver Lesion Classification.” *Neurocomputing*, vol. 321, Dec. 2018, pp. 321–331, <https://doi.org/10.1016/j.neucom.2018.09.013>.
- [24] Karp, Rafal, and Zaneta Swiderska-Chadaj. Automatic Generation of Graphical Game Assets Using GAN. 13 July 2021, <https://doi.org/10.1145/3477911.3477913>.
- [25] L. Jiao and J. Zhao, "A Survey on the New Generation of Deep Learning in Image Processing," in *IEEE Access*, vol. 7, pp. 172231-172263, 2019, doi: 10.1109/ACCESS.2019.2956508.
- [26] L. Wang, W. Chen, W. Yang, F. Bi and F. R. Yu, "A State-of-the-Art Review on Image Synthesis With Generative Adversarial Networks," in *IEEE Access*, vol. 8, pp. 63514-63537, 2020, doi: 10.1109/ACCESS.2020.2982224.
- [27] Shorten, Connor, and Taghi M. Khoshgoftaar. “A Survey on Image Data Augmentation for Deep Learning.” *Journal of Big Data*, vol. 6, no. 1, 6 July 2019, [journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0](http://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0), <https://doi.org/10.1186/s40537-019-0197-0>.
- [28] Kayalibay, Baris, et al. “CNN-Based Segmentation of Medical Imaging Data.” *ArXiv:1701.03056 [Cs]*, 25 July 2017, [arxiv.org/abs/1701.03056](http://arxiv.org/abs/1701.03056).

- [29] Jain, M., & Shah, A. (2022). Machine Learning with Convolutional Neural Networks (CNNs) in Seismology for Earthquake Prediction. *Iconic Research and Engineering Journals*, 5(8), 389–398. <https://www.irejournals.com/paper-details/1707057>
- [30] Kaushik, P., & Jain, M. A Low Power SRAM Cell for High Speed Applications Using 90nm Technology. *Csjournals. Com*, 10. <https://www.csjournals.com/IJEE/PDF10-2/66.%20Puneet.pdf>
- [31] Kaushik, P., & Jain, M. (2018). Design of low power CMOS low pass filter for biomedical application. *International Journal of Electrical Engineering & Technology (IJEET)*, 9(5).
- [32] Kumar, Y., Saini, S., & Payal, R. (2020). Comparative Analysis for Fraud Detection Using Logistic Regression, Random Forest and Support Vector Machine. *SSRN Electronic Journal*.
- [33] Höppner, S., Baesens, B., Verbeke, W., & Verdonck, T. (2020). Instance-Dependent Cost-Sensitive Learning for Detecting Transfer Fraud. *arXiv preprint arXiv:2005.02488*.
- [34] Niu, X., Wang, L., & Yang, X. (2019). A Comparison Study of Credit Card Fraud Detection: Supervised versus Unsupervised. *arXiv preprint arXiv:1904.10604*.
- [35] Bhat, N. (2019). Fraud detection: Feature selection-over sampling. *Kaggle*. Retrieved from <https://www.kaggle.com/code/nareshbhat/fraud-detection-feature-selection-over-sampling>
- [36] InsiderFinance Wire. (2021). Logistic regression: A simple powerhouse in fraud detection. *Medium*. Retrieved from <https://wire.insiderfinance.io/logistic-regression-a-simple-powerhouse-in-fraud-detection-15ab984b2102>
- [37] Olaitan, V. O. (2020). Feature-based selection technique for credit card fraud detection. Master's Thesis, National College of Ireland. Retrieved from <https://norma.ncirl.ie/5122/1/olaitanvictoriaolanlokun.pdf>
- [38] Raymaekers, J., Verbeke, W., & Verdonck, T. (2021). Weight-of-evidence 2.0 with shrinkage and spline-binning. *arXiv preprint arXiv:2101.01494*. Retrieved from <https://arxiv.org/abs/2101.01494>
- [38] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2017). Credit card fraud detection: A realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8), 3784–3797. <https://doi.org/10.1109/TNNLS.2017.2736643>
- [40] Carcillo, F., Dal Pozzolo, A., Le Borgne, Y. A., Caelen, O., Mazzer, Y., & Bontempi, G. (2019). Scarff: A scalable framework for streaming credit card fraud detection with spark. *Information Fusion*, 41, 182–194. <https://doi.org/10.1016/j.inffus.2017.09.005>
- [41] West, J., & Bhattacharya, M. (2016). Intelligent financial fraud detection: A comprehensive review. *Computers & Security*, 57, 47–66. <https://doi.org/10.1016/j.cose.2015.09.005>
- [42] Zareapoor, M., & Shamsolmoali, P. (2015). Application of credit card fraud detection: Based on bagging ensemble classifier. *Procedia Computer Science*, 48, 679–685. <https://doi.org/10.1016/j.procs.2015.04.201>
- [43] Patel, H., & Zaveri, M. (2011). Credit card fraud detection using neural network. *International Journal of Innovative Research in Computer and Communication Engineering*, 1(2), 1–6. [https://www.ijircce.com/upload/2011/october/1\\_Credit.pdf](https://www.ijircce.com/upload/2011/october/1_Credit.pdf)
- [44] Puneet Kaushik, Mohit Jain, Gayatri Patidar, Paradayil Rhea Eapen, Chandra Prabha Sharma (2018). Smart Floor Cleaning Robot Using Android. *International Journal of Electronics Engineering*. <https://www.csjournals.com/IJEE/PDF10-2/64.%20Puneet.pdf>
- [45] Duman, E., & Ozcelik, M. H. (2011). Detecting credit card fraud by genetic algorithm and scatter search. *Expert Systems with Applications*, 38(10), 13057–13063. <https://doi.org/10.1016/j.eswa.2011.04.102>
- [46] Puneet Kaushik, Mohit Jain. “A Low Power SRAM Cell for High Speed Applications Using 90nm Technology.” *Csjournals.Com* 10, no. 2 (December 2018):

- 6.<https://www.csjournals.com/IJEE/PDF10-2/66.%20Puneet.pdf>
- [47] Jain, M., & Srihari, A. (2021). Comparison of CAD detection of mammogram with SVM and CNN. IRE Journals, 8(6), 63-75. <https://www.irejournals.com/formatedpaper/1706647.pdf>
- [48] Kaushik, P., Jain, M., & Jain, A. (2018). A pixel-based digital medical images protection using genetic algorithm. International Journal of Electronics and Communication Engineering, 31-37. [http://www.irphouse.com/ijece18/ijecev11n1\\_05.pdf](http://www.irphouse.com/ijece18/ijecev11n1_05.pdf)
- [49] Kaushik, P., Jain, M., & Shah, A. (2018). A Low Power Low Voltage CMOS Based Operational Transconductance Amplifier for Biomedical Application. <https://ijsetr.com/uploads/136245IJSETR17012-283.pdf>
- [50] Jain, M., & Shah, A. (2022). Machine Learning with Convolutional Neural Networks (CNNs) in Seismology for Earthquake Prediction. Iconic Research and Engineering Journals, 5(8), 389–398. <https://www.irejournals.com/paper-details/1707057>
- [51] Kaushik, P., & Jain, M. (2018). Design of low power CMOS low pass filter for biomedical application. International Journal of Electrical Engineering & Technology (IJEET), 9(5).