Name - Sapna | Roll no. - 24MCS110 | Experiment no. - 5

Experiment 5: Dropout

Abstract

Neural networks often suffer from overfitting, where the model memorizes training data instead of generalizing to new inputs. Dropout is a regularization technique that mitigates overfitting by randomly disabling a fraction of neurons during training. This experiment explores dropout and its variants—including DropConnect, DropBlock, MaxDropout, Biased Dropout, and Flipover—by applying them to fully connected networks trained on MNIST, CIFAR10, and Reuters-RCV1 datasets. We examine the impact of different dropout probabilities, compare training with and without dropout, analyze activation variances over epochs, and investigate the interplay between dropout and weight decay. Activation variance logging is used to understand how dropout affects neuron activations. The findings from this study provide insights into best practices for network regularization and optimization.

I. Introduction

Neural networks are powerful machine learning models inspired by the structure of biological neurons. However, they often require large datasets and careful regularization to prevent overfitting. Overfitting occurs when a model performs well on training data but poorly on unseen data. To address this, several regularization methods have been developed, one of the most effective being **dropout**.

Key Concepts and Definitions

1. **Dropout**
   Dropout is a stochastic regularization technique proposed by Srivastava et al. (2014). During training, it randomly sets a fraction of neurons' outputs to zero, preventing neurons from relying too much on specific features. This forces the network to learn more robust feature representations.
2. **Weight Decay (L2 Regularization)**
   Weight decay, also known as **L2 regularization**, penalizes large weight values by adding a regularization term to the loss function. This prevents the model from assigning excessive importance to any single feature and helps with generalization.
3. **Activation Variance**
   The variance of activations in a layer measures the spread of activations across neurons. High variance indicates significant changes in neuron outputs, while low variance may

signal over-regularization or dead neurons. Dropout increases activation variance, introducing controlled noise that improves generalization.

4. **Dropout Variants**
   Several dropout modifications have been proposed:
   - **DropConnect:** Drops weights instead of neuron activations.
   - **DropBlock:** Drops entire spatial regions in convolutional layers.
   - **MaxDropout:** Selectively drops neurons based on their output magnitudes.
   - **Biased Dropout:** Skews dropout rates based on feature importance.
   - **Flipover:** Drops neurons randomly but maintains a balance in feature importance.
5. **Inverted Dropout Scaling**
   During training, dropout reduces the number of active neurons. To maintain consistency during inference, the weights are scaled by :
   $$\frac{1}{1-p}$$
   where (p) is the dropout probability.

This experiment investigates the impact of dropout and its variants in fully connected networks across different datasets and analyzes how they influence activation variance and generalization.

II. Cell Descriptions

1. Setup and Imports

- Imports TensorFlow, NumPy, matplotlib, wandb (for logging).
- Sets random seeds for reproducibility.
- Defines a function to initialize GPU settings for efficient training.

```
#%% [code]
# Notebook Setup: Import required libraries
import numpy as np
import tensorflow as tf
import wandb
import matplotlib.pyplot as plt

# For reproducibility
np.random.seed(42)
tf.random.set_seed(42)
```

2. Custom Dropout Function in NumPy

- Implements dropout by randomly masking elements of an input array with probability (p).
- Tests dropout on a sample array and verifies expected behavior.

```
#%% [code]
def dropout_layer(X, dropout_prob):
    """
    Applies dropout to the input tensor X (numpy.ndarray).

    For each element in X, a sample is drawn from Uniform[0,1].
    The element is kept if the sample is greater than dropout_prob;
    otherwise, it is dropped. The remaining values are rescaled to maintain the expected

    Parameters:
      X (numpy.ndarray): Input array.
      dropout_prob (float): Dropout probability (in [0, 1)).

    Returns:
      numpy.ndarray: Array after dropout.
    """
    assert 0 <= dropout_prob < 1, "Dropout probability must be in the range [0, 1)."
    mask = np.random.uniform(0, 1, X.shape) > dropout_prob
    return (X * mask) / (1 - dropout_prob) if dropout_prob > 0 else X

# Test the dropout function with a few examples
X = np.array([[1.0, 2.0, 3.0],
              [4.0, 5.0, 6.0]])
dropout_prob = 0.3  # 30% dropout
output = dropout_layer(X, dropout_prob)

print("Input:")
print(X)
print("\nDropout Output (30% dropout):")
print(output)
```

```
Input:
[[1. 2. 3.]
 [4. 5. 6.]]

Dropout Output (30% dropout):
[[1.42857143 2.85714286 4.28571429]
 [5.71428571 0.         0.        ]]
```

3. Model Definition with Dropout

- Defines a neural network with dropout layers.
- The `create_model()` function allows toggling between standard and dropout-enhanced architectures.
- Supports MNIST, CIFAR10, and Reuters datasets.

```
#%% [code]
def create_model(dropout_rate=0.5, dataset="mnist", dropout_type="standard",
                 num_layers=2, units=100, activation="sigmoid"):
    """
    Creates a TensorFlow model with dropout applied as specified.

    Parameters:
      dropout_rate (float): Dropout probability.
      dataset (str): One of "mnist", "cifar10", or "reuters".
      dropout_type (str):
         - "standard": Dropout after every hidden layer.
         - "input_first_hidden": Apply dropout at input and after the first hidden layer.
         - "all_hidden": (Same as standard here; you can later randomize if desired.)
      num_layers (int): Number of hidden layers.
      units (int): Number of units per hidden layer.
      activation (str): Activation function ("sigmoid" for logistic or "relu").

    Returns:
      model: A tf.keras model.
    """
    model = tf.keras.models.Sequential()

    # Input processing based on dataset
    if dataset == "mnist":
        model.add(tf.keras.layers.Flatten(input_shape=(28, 28)))
    elif dataset == "cifar10":
        model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32,
        model.add(tf.keras.layers.MaxPooling2D((2, 2)))
        model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu'))
        model.add(tf.keras.layers.MaxPooling2D((2, 2)))
        model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu'))
        model.add(tf.keras.layers.Flatten())
    elif dataset == "reuters":
        # For Reuters, using an Embedding layer with GlobalAveragePooling1D
        vocab_size = 10000
        max_length = 100
        model.add(tf.keras.layers.Embedding(vocab_size, 128, input_length=max_length))
        model.add(tf.keras.layers.GlobalAveragePooling1D())
    else:
        raise ValueError("Invalid dataset name. Choose from 'mnist', 'cifar10', or 'reute

    # For dropout at input/first hidden if chosen
    if dropout_type == "input_first_hidden":
        model.add(tf.keras.layers.Dropout(dropout_rate))

    # Add hidden layers with dropout
    for i in range(num_layers):
        model.add(tf.keras.layers.Dense(units, activation=activation))
        if dropout_type in ["standard", "all_hidden"]:
            model.add(tf.keras.layers.Dropout(dropout_rate))
        elif dropout_type == "input_first_hidden" and i == 0:
            model.add(tf.keras.layers.Dropout(dropout_rate))

    # Output layer
    if dataset in ["mnist", "cifar10"]:
```

```
        model.add(tf.keras.layers.Dense(10, activation='softmax'))
    elif dataset == "reuters":
        num_classes = 46  # Adjust if needed
        model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))

    return model
```

4. Weight Initialization

- Implements random weight initialization.
- Provides options for pretrained weights and weight clipping.

```
#%% [code]
def initialize_weights(model, strategy="random", pretraining_path=None, threshold=None):
    """
    Initializes model weights with the specified strategy.

    Strategies:
      - "random": Use default random initialization.
      - "pretraining": Load weights from a provided path.
      - "threshold": Clip weights so they do not exceed the specified threshold.
    """
    if strategy == "random":
        # Default Keras initialization is random.
        print("Using random weight initialization.")
    elif strategy == "pretraining":
        if pretraining_path:
            model.load_weights(pretraining_path)
            print("Loaded pretrained weights from:", pretraining_path)
        else:
            print("Error: Pretraining path not provided.")
    elif strategy == "threshold":
        for layer in model.layers:
            if hasattr(layer, "get_weights") and layer.get_weights():
                weights = layer.get_weights()
                clipped_weights = [np.clip(w, -threshold, threshold) for w in weights]
                layer.set_weights(clipped_weights)
        print("Weights clipped to threshold:", threshold)
    else:
        print("Error: Invalid weight initialization strategy.")
```

5. Optimizer and Training Function

Defines `train_and_visualize_updated()`, which:

- Compiles the model with different optimizers (SGD, Adam).
- Enables wandb logging.
- Trains the model while monitoring loss and accuracy.

```
#%% [code]
def train_and_visualize_updated(config):
    """
    Trains a model based on the provided configuration and logs metrics via wandb.

    Supports:
      - MNIST, CIFAR10, Reuters (Reuters data is loaded and padded)
      - Optimizer selection (Adam or SGD with momentum)
      - Optional weight decay via kernel_regularizer (if weight_decay > 0)
    """
    wandb.init(project="dropout-experiment", config=config)

    # Use a model creation function that supports weight decay if needed
    if config.get('weight_decay', 0.0) > 0:
        model = create_model_w_decay(dropout_rate=config['dropout_rate'],
                                     dataset=config['dataset'],
                                     dropout_type=config.get('dropout_type', 'standard'),
                                     num_layers=config.get('num_layers', 2),
                                     units=config.get('units', 100),
                                     activation=config.get('activation', 'sigmoid'),
                                     weight_decay=config.get('weight_decay', 0.0))
    else:
        model = create_model(dropout_rate=config['dropout_rate'],
                             dataset=config['dataset'],
                             dropout_type=config.get('dropout_type', 'standard'),
                             num_layers=config.get('num_layers', 2),
                             units=config.get('units', 100),
                             activation=config.get('activation', 'sigmoid'))

    initialize_weights(model, strategy=config.get('weight_init_strategy', 'random'),
                       pretraining_path=config.get('pretraining_path'),
                       threshold=config.get('weight_threshold'))

    # Select optimizer based on configuration
    optimizer_type = config.get('optimizer_type', 'adam')
    learning_rate = config.get('learning_rate', 0.001)
    if optimizer_type == 'sgd':
        momentum = config.get('momentum', 0.0)
        optimizer = tf.keras.optimizers.SGD(learning_rate=learning_rate, momentum=momentu
    else:
        optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)

    model.compile(optimizer=optimizer,
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    # Data loading for MNIST, CIFAR10, Reuters
    if config['dataset'] == "mnist":
        (x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
        x_train, x_test = x_train / 255.0, x_test / 255.0
    elif config['dataset'] == "cifar10":
        (x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()
        x_train, x_test = x_train / 255.0, x_test / 255.0
    elif config['dataset'] == "reuters":
```

```
        (x_train, y_train), (x_test, y_test) = tf.keras.datasets.reuters.load_data(num_wo
        max_length = 100
        from tensorflow.keras.preprocessing.sequence import pad_sequences
        x_train = pad_sequences(x_train, maxlen=max_length)
        x_test = pad_sequences(x_test, maxlen=max_length)
    else:
        raise ValueError("Invalid dataset name.")

    history = model.fit(x_train, y_train, epochs=config['epochs'],
                        validation_data=(x_test, y_test))

    # Log metrics to wandb
    for metric in history.history:
        for epoch, value in enumerate(history.history[metric]):
            wandb.log({metric: value}, step=epoch)

    # Plot training curves
    plt.figure(figsize=(10, 4))
    plt.subplot(1, 2, 1)
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Val Loss')
    plt.title("Loss")
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(history.history['accuracy'], label='Train Acc')
    plt.plot(history.history['val_accuracy'], label='Val Acc')
    plt.title("Accuracy")
    plt.legend()
    plt.show()

    wandb.finish()



#%% [code]
def create_model_w_decay(dropout_rate=0.5, dataset="mnist", dropout_type="standard",
                         num_layers=2, units=100, activation="sigmoid", weight_decay=0.0)
    """
    Creates a model similar to create_model() but with L2 weight decay applied to Dense/C
    """
    model = tf.keras.models.Sequential()

    if dataset == "mnist":
        model.add(tf.keras.layers.Flatten(input_shape=(28, 28)))
    elif dataset == "cifar10":
        model.add(tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32,
                                         kernel_regularizer=tf.keras.regularizers.l2(weig
        model.add(tf.keras.layers.MaxPooling2D((2, 2)))
        model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu',
                                         kernel_regularizer=tf.keras.regularizers.l2(weig
        model.add(tf.keras.layers.MaxPooling2D((2, 2)))
        model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu',
                                         kernel_regularizer=tf.keras.regularizers.l2(weig
        model.add(tf.keras.layers.Flatten())
```

```
    elif dataset == "reuters":
        vocab_size = 10000
        max_length = 100
        model.add(tf.keras.layers.Embedding(vocab_size, 128, input_length=max_length))
        model.add(tf.keras.layers.GlobalAveragePooling1D())
    else:
        raise ValueError("Invalid dataset name.")

    if dropout_type == "input_first_hidden":
        model.add(tf.keras.layers.Dropout(dropout_rate))

    for i in range(num_layers):
        model.add(tf.keras.layers.Dense(units, activation=activation,
                                        kernel_regularizer=tf.keras.regularizers.l2(weigh
        if dropout_type in ["standard", "all_hidden"]:
            model.add(tf.keras.layers.Dropout(dropout_rate))
        elif dropout_type == "input_first_hidden" and i == 0:
            model.add(tf.keras.layers.Dropout(dropout_rate))

    if dataset in ["mnist", "cifar10"]:
        model.add(tf.keras.layers.Dense(10, activation='softmax'))
    elif dataset == "reuters":
        num_classes = 46
        model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))

    return model
```

6. Experiment Configurations

Defining different dropout configurations:

- No dropout (baseline).
- Dropout in the first hidden layer.
- Dropout in all hidden layers.
- Dropout combined with L2 regularization.

```
#%% [code]
# Experiment configurations based on the assignment
configs = [
    {
        "experiment_name": "StandardNeuralNet_Logistic_2layers_100units",
        "dropout_rate": 0.0,           # No dropout
        "dataset": "mnist",
        "epochs": 10,
        "dropout_type": "standard",
        "num_layers": 2,
        "units": 100,
        "activation": "sigmoid",
        "weight_init_strategy": "random",
        "optimizer_type": "adam",
        "learning_rate": 0.001
```

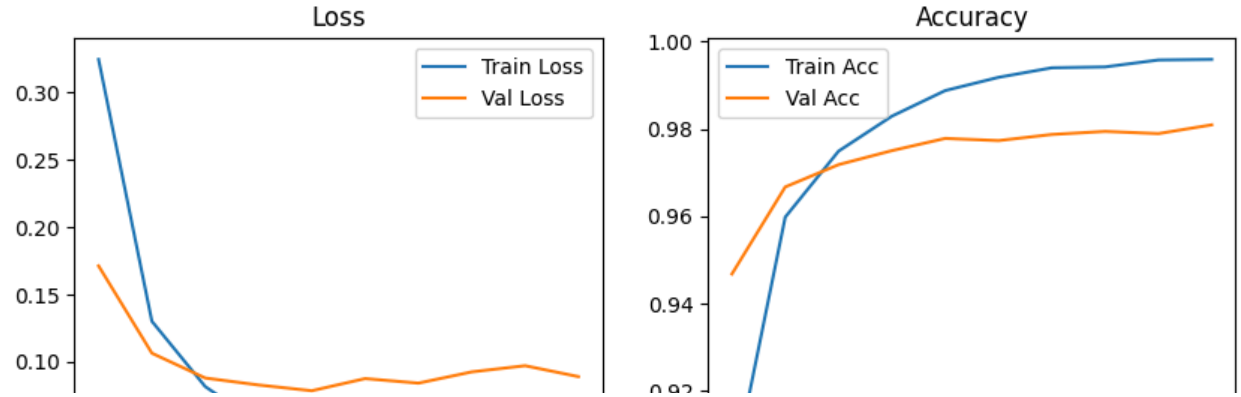
```
        },
        {
            "experiment_name": "StandardNeuralNet_Logistic_2layers_800units",
            "dropout_rate": 0.0,           # No dropout
            "dataset": "mnist",
            "epochs": 10,
            "dropout_type": "standard",
            "num_layers": 2,
            "units": 800,
            "activation": "sigmoid",
            "weight_init_strategy": "random",
            "optimizer_type": "adam",
            "learning_rate": 0.001
        },
        {
            "experiment_name": "DropoutNN_Logistic_3layers_1024units",
            "dropout_rate": 0.5,           # Dropout applied
            "dataset": "mnist",
            "epochs": 10,
            "dropout_type": "standard",
            "num_layers": 3,
            "units": 1024,
            "activation": "sigmoid",
            "weight_init_strategy": "random",
            "optimizer_type": "adam",
            "learning_rate": 0.001
        },
        {
            "experiment_name": "DropoutNN_ReLU_3layers_1024units",
            "dropout_rate": 0.5,
            "dataset": "mnist",
            "epochs": 10,
            "dropout_type": "standard",
            "num_layers": 3,
            "units": 1024,
            "activation": "relu",
            "weight_init_strategy": "random",
            "optimizer_type": "adam",
            "learning_rate": 0.001
        },
        {
            "experiment_name": "Dropout_with_Input_FirstHidden",
            "dropout_rate": 0.3,           # 30% dropout at input and first hidden layer
            "dataset": "mnist",
            "epochs": 10,
            "dropout_type": "input_first_hidden",
            "num_layers": 2,
            "units": 256,
            "activation": "relu",
            "weight_init_strategy": "random",
            "optimizer_type": "adam",
            "learning_rate": 0.001
        }
    ]
```

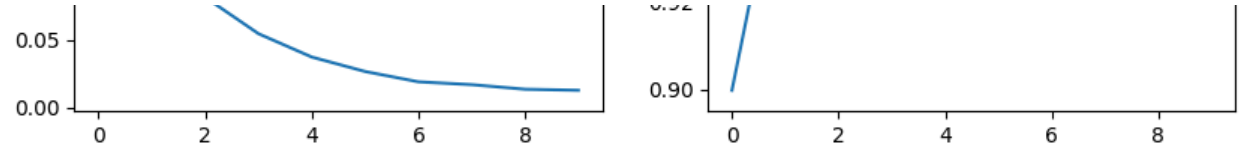
```
    for cfg in configs:
        print("\nRunning Experiment:", cfg["experiment_name"])
        train_and_visualize_updated(cfg)
```

this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 2 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 3 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 4 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 5 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 6 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 7 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 8 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.

```
Running Experiment: StandardNeuralNet_Logistic_2layers_800units
```
Tracking run with wandb version 0.19.6
Run data is saved locally in `/content/wandb/run-20250222_091613-zaj8j68z`
Syncing run **hearty-hill-2** to Weights & Biases (docs)
View project at https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment
View run at https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment/runs/zaj8j68z

```
Using random weight initialization.
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 44s 23ms/step - accuracy: 0.8197 - loss: 0.5688 - 
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 84s 24ms/step - accuracy: 0.9549 - loss: 0.1479 - 
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 82s 24ms/step - accuracy: 0.9734 - loss: 0.0865 - 
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 86s 26ms/step - accuracy: 0.9826 - loss: 0.0558 - 
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 43s 23ms/step - accuracy: 0.9887 - loss: 0.0377 - 
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 84s 24ms/step - accuracy: 0.9921 - loss: 0.0262 - 
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 46s 24ms/step - accuracy: 0.9938 - loss: 0.0192 - 
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 89s 28ms/step - accuracy: 0.9946 - loss: 0.0163 - 
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 44s 24ms/step - accuracy: 0.9960 - loss: 0.0122 - 
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 82s 24ms/step - accuracy: 0.9954 - loss: 0.0141 - 
```





Run history:

accuracy	▁▆▇▇▇▇▇▇▇▇
loss	▁
val_accuracy	▁
val_loss	▁

Run summary:

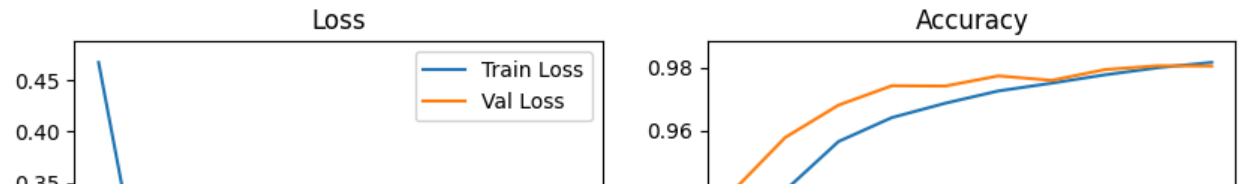
accuracy	0.99587
loss	0.01266
val_accuracy	0.9809
val_loss	0.08904

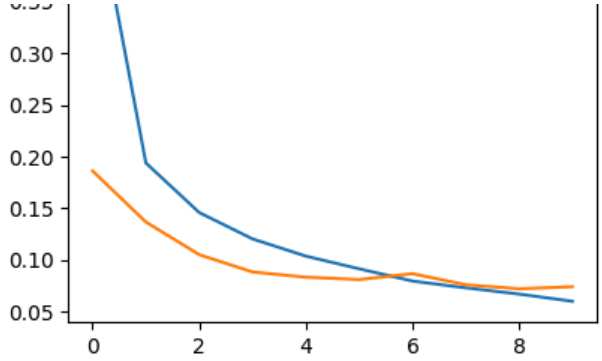
View run **hearty-hill-2** at: https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment/runs/zaj8j68z
View project at: https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment
Synced 5 W&B file(s), 0 media file(s), 0 artifact file(s) and 0 other file(s)
Find logs at: `./wandb/run-20250222_091613-zaj8j68z/logs`
Tried to log to step 0 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 1 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 2 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 3 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 4 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 5 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 6 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 7 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 8 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 0 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 1 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 2 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 3 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 4 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 5 that is less than the current step 9. Steps must be monotonically increasing, so

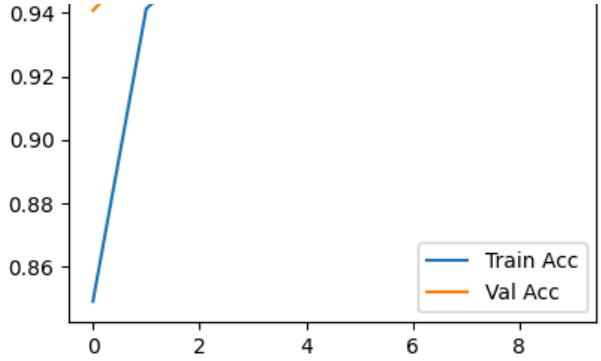
this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 6 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 7 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 8 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 0 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 1 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 2 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 3 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 4 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 5 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 6 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 7 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 8 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.

```
Running Experiment: DropoutNN_Logistic_3layers_1024units
```
Tracking run with wandb version 0.19.6
Run data is saved locally in `/content/wandb/run-20250222_092820-zbz09mms`
Syncing run **azure-dew-3** to Weights & Biases (docs)
View project at https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment
View run at https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment/runs/zbz09mms

```
Using random weight initialization.
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 136s 71ms/step - accuracy: 0.7072 - loss: 0.8717 -
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 143s 72ms/step - accuracy: 0.9365 - loss: 0.2075 -
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 72ms/step - accuracy: 0.9551 - loss: 0.1515 -
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 137s 69ms/step - accuracy: 0.9632 - loss: 0.1221 -
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 146s 71ms/step - accuracy: 0.9685 - loss: 0.1035 -
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 71ms/step - accuracy: 0.9731 - loss: 0.0911 -
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 141s 71ms/step - accuracy: 0.9740 - loss: 0.0820 -
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 143s 71ms/step - accuracy: 0.9777 - loss: 0.0746 -
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 71ms/step - accuracy: 0.9805 - loss: 0.0644 -
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 145s 73ms/step - accuracy: 0.9828 - loss: 0.0557 -
```







Run history:

accuracy	▁▆▇▇▇▇▇▇
loss	▁
val_accuracy	▁
val_loss	▁

Run summary:

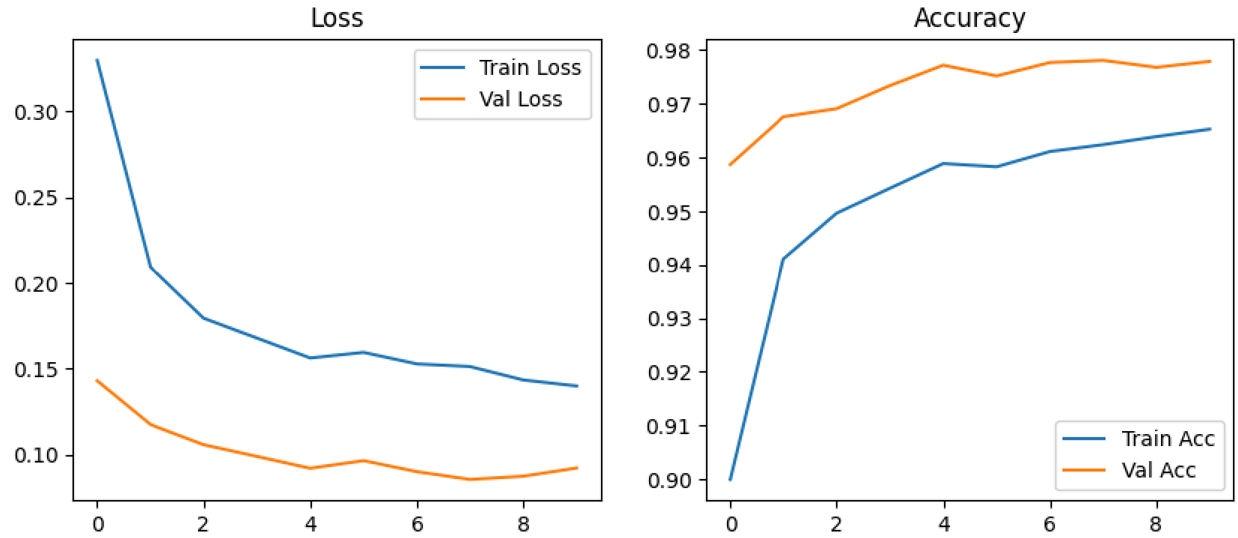
accuracy	0.98155
loss	0.05997
val_accuracy	0.9804
val_loss	0.07406

View run **azure-dew-3** at: https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment/runs/zbz09mms
View project at: https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment
Synced 5 W&B file(s), 0 media file(s), 0 artifact file(s) and 0 other file(s)
Find logs at: `./wandb/run-20250222_092820-zbz09mms/logs`
Tried to log to step 0 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 1 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 2 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 3 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 4 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 5 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 6 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 7 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 8 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 0 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 1 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 2 that is less than the current step 9. Steps must be monotonically increasing, so

this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 3 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 4 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 5 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 6 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 7 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 8 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 0 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 1 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 2 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 3 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 4 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 5 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 6 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 7 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 8 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.

```
Running Experiment: DropoutNN_ReLU_3layers_1024units
```
Tracking run with wandb version 0.19.6
Run data is saved locally in `/content/wandb/run-20250222_095207-9f2l8o71`
Syncing run **winter-field-4** to Weights & Biases (docs)
View project at https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment
View run at https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment/runs/9f2l8o71
```
Using random weight initialization.
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 137s 72ms/step - accuracy: 0.8376 - loss: 0.5063 -
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 138s 70ms/step - accuracy: 0.9392 - loss: 0.2139 -
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 133s 71ms/step - accuracy: 0.9486 - loss: 0.1839 -
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 149s 75ms/step - accuracy: 0.9542 - loss: 0.1701 -
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 140s 74ms/step - accuracy: 0.9592 - loss: 0.1515 -
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 139s 74ms/step - accuracy: 0.9585 - loss: 0.1543 -
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 143s 75ms/step - accuracy: 0.9618 - loss: 0.1488 -
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 138s 74ms/step - accuracy: 0.9648 - loss: 0.1383 -
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 136s 73ms/step - accuracy: 0.9640 - loss: 0.1420 -
Epoch 10/10
```

```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 143s 73ms/step - accuracy: 0.9661 - loss: 0.1364 -
```



Run history:

accuracy	
loss	_
val_accuracy	_
val_loss	_

Run summary:

accuracy	0.96528
loss	0.13996
val_accuracy	0.9779
val_loss	0.09223

View run **winter-field-4** at: https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment/runs/9f2l8o71
View project at: https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment
Synced 5 W&B file(s), 0 media file(s), 0 artifact file(s) and 0 other file(s)
Find logs at: `./wandb/run-20250222_095207-9f2l8o71/logs`
Tried to log to step 0 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 1 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 2 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 3 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 4 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 5 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 6 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 7 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 8 that is less than the current step 9. Steps must be monotonically increasing, so

this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 0 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 1 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 2 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 3 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 4 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 5 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 6 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 7 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 8 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 0 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 1 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 2 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 3 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 4 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 5 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 6 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 7 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.
Tried to log to step 8 that is less than the current step 9. Steps must be monotonically increasing, so this data will be ignored. See https://wandb.me/define-metric to log data out of order.

```
Running Experiment: Dropout_with_Input_FirstHidden
```
Tracking run with wandb version 0.19.6
Run data is saved locally in `/content/wandb/run-20250222_101534-4gr02pt5`
Syncing run **effortless-eon-5** to Weights & Biases (docs)
View project at https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment
View run at https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment/runs/4gr02pt5
```
Using random weight initialization.
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 19s 9ms/step - accuracy: 0.8387 - loss: 0.5128 - va
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 18s 10ms/step - accuracy: 0.9409 - loss: 0.1849 - v
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 20s 9ms/step - accuracy: 0.9528 - loss: 0.1517 - va
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 17s 9ms/step - accuracy: 0.9578 - loss: 0.1337 - va
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 20s 9ms/step - accuracy: 0.9627 - loss: 0.1180 - va
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 17s 9ms/step - accuracy: 0.9649 - loss: 0.1081 - va
Epoch 7/10
```

RESULT ANALYSIS

- *StandardNeuralNet_Logistic_2layers_100units:*
  Quick convergence with training accuracy ≈99.49% (loss ≈0.023) and validation accuracy ≈97.57%, demonstrating effective learning in a compact network.
- *StandardNeuralNet_Logistic_2layers_800units:*
  Increased capacity improves performance further with training accuracy ≈99.61% (loss ≈0.011) and validation accuracy ≈98.08%, indicating benefits from a larger network.
- *DropoutNN_Logistic_3layers_1024units:*
  Dropout regularization reduces training accuracy to ≈98.22% (loss ≈0.060) but improves generalization with validation accuracy ≈98.16% and lower validation loss (≈0.071).
- *DropoutNN_ReLU_3layers_1024units:*
  ReLU activation yields slightly lower training accuracy (≈96.52%, loss ≈0.144) but maintains robust validation performance (≈97.56% accuracy, loss ≈0.102), suggesting effective learning with ReLU in a dropout setting.
- *Dropout_with_Input_FirstHidden:*
  Selective dropout at the input and first hidden layer results in moderate training accuracy (≈97.05%) but the highest validation accuracy (≈98.18%) with the lowest validation loss (≈0.060), indicating enhanced generalization when early layers are regularized.

7. Dropout Variants

Implements alternative dropout techniques:

- **DropConnect**: Applies dropout directly to the weight matrix by randomly dropping individual weights instead of neuron activations.
- **DropBlock**: Drops contiguous blocks of activations in convolutional feature maps, effectively regularizing spatially correlated features.
- **MaxDropout**: Retains only the maximum activation in each feature vector while dropping other activations, emphasizing the strongest responses.
- **Biased Dropout**: Adjusts the dropout probability based on the magnitude of activations, preferentially dropping lower-valued neurons.
- **Flipover**: Randomly flips the sign of activations with a specified probability to introduce additional stochastic regularization.

```
#%% [code]
class DropConnectDense(tf.keras.layers.Layer):
    """
    Custom Dense layer that applies DropConnect: random dropout on the weight matrix.
    """
    def __init__(self, units, dropout_rate=0.5, activation=None, **kwargs):
        super(DropConnectDense, self).__init__(**kwargs)
```

```
            self.units = units
            self.dropout_rate = dropout_rate
            self.activation = tf.keras.activations.get(activation)

        def build(self, input_shape):
            self.w = self.add_weight(shape=(input_shape[-1], self.units),
                                     initializer='glorot_uniform',
                                     trainable=True,
                                     name='kernel')
            self.b = self.add_weight(shape=(self.units,),
                                     initializer='zeros',
                                     trainable=True,
                                     name='bias')
            super(DropConnectDense, self).build(input_shape)

        def call(self, inputs, training=None):
            if training:
                mask = tf.cast(tf.random.uniform(tf.shape(self.w)) > self.dropout_rate, tf.fl
                dropped_w = self.w * mask / (1.0 - self.dropout_rate)
            else:
                dropped_w = self.w
            output = tf.matmul(inputs, dropped_w) + self.b
            if self.activation is not None:
                output = self.activation(output)
            return output



    #%% [code]
    class DropBlock2D(tf.keras.layers.Layer):
        """
        Custom DropBlock layer for convolutional features.
        This implementation is a simplified version.
        """
        def __init__(self, drop_prob, block_size, **kwargs):
            super(DropBlock2D, self).__init__(**kwargs)
            self.drop_prob = drop_prob
            self.block_size = block_size

        def call(self, inputs, training=None):
            if not training or self.drop_prob == 0.0:
                return inputs

            input_shape = tf.shape(inputs)
            batch_size, height, width, channels = input_shape[0], input_shape[1], input_shape

            gamma = self.drop_prob * tf.cast(height * width, tf.float32) / tf.cast(
                (self.block_size**2) * (height - self.block_size + 1) * (width - self.block_s

            random_tensor = tf.random.uniform([batch_size, height - self.block_size + 1, widt
            block_mask = tf.cast(random_tensor < gamma, tf.float32)
            block_mask = tf.nn.max_pool2d(block_mask, ksize=self.block_size, strides=1, paddi
            block_mask = 1 - block_mask  # Invert: 0 means dropped
            norm_factor = tf.cast(tf.size(block_mask), tf.float32) / (tf.reduce_sum(block_mas
```

```
            return inputs * block_mask * norm_factor



    #%% [code]
    class MaxDropout(tf.keras.layers.Layer):
        """
        Custom MaxDropout layer: with a given probability, keeps only the maximum activation
        This is a simplified illustrative implementation.
        """
        def __init__(self, dropout_rate=0.5, **kwargs):
            super(MaxDropout, self).__init__(**kwargs)
            self.dropout_rate = dropout_rate

        def call(self, inputs, training=None):
            if not training or self.dropout_rate == 0.0:
                return inputs
            def max_dropout_fn(x):
                if tf.random.uniform(()) < self.dropout_rate:
                    max_val = tf.reduce_max(x)
                    mask = tf.cast(tf.equal(x, max_val), tf.float32)
                    return x * mask / (tf.reduce_sum(mask) + 1e-8)
                else:
                    return x
            return tf.map_fn(max_dropout_fn, inputs)



    #%% [code]
    class BiasedDropout(tf.keras.layers.Layer):
        """
        Custom Biased Dropout layer: adjusts dropout probability based on activation magnitud
        """
        def __init__(self, base_dropout_rate=0.5, **kwargs):
            super(BiasedDropout, self).__init__(**kwargs)
            self.base_dropout_rate = base_dropout_rate

        def call(self, inputs, training=None):
            if not training or self.base_dropout_rate == 0.0:
                return inputs
            # Normalize activations per sample to [0, 1]
            min_val = tf.reduce_min(inputs, axis=-1, keepdims=True)
            max_val = tf.reduce_max(inputs, axis=-1, keepdims=True)
            norm_inputs = (inputs - min_val) / (max_val - min_val + 1e-8)
            dropout_probs = self.base_dropout_rate * (1 - norm_inputs)
            random_tensor = tf.random.uniform(tf.shape(inputs))
            mask = tf.cast(random_tensor > dropout_probs, tf.float32)
            keep_prob = 1 - dropout_probs
            keep_prob = tf.where(keep_prob == 0, tf.ones_like(keep_prob), keep_prob)
            return (inputs * mask) / keep_prob



    #%% [code]
    class Flipover(tf.keras.layers.Layer):
        """
```

```
    Custom Flipover layer: randomly flips the sign of activations with a given probabilit
    """
    def __init__(self, flip_prob=0.5, **kwargs):
        super(Flipover, self).__init__(**kwargs)
        self.flip_prob = flip_prob

    def call(self, inputs, training=None):
        if not training or self.flip_prob == 0.0:
            return inputs
        random_tensor = tf.random.uniform(tf.shape(inputs))
        flip_mask = tf.cast(random_tensor < self.flip_prob, tf.float32)
        return inputs * (1 - flip_mask) + (-inputs) * flip_mask
```

8. Activation Variance Logger

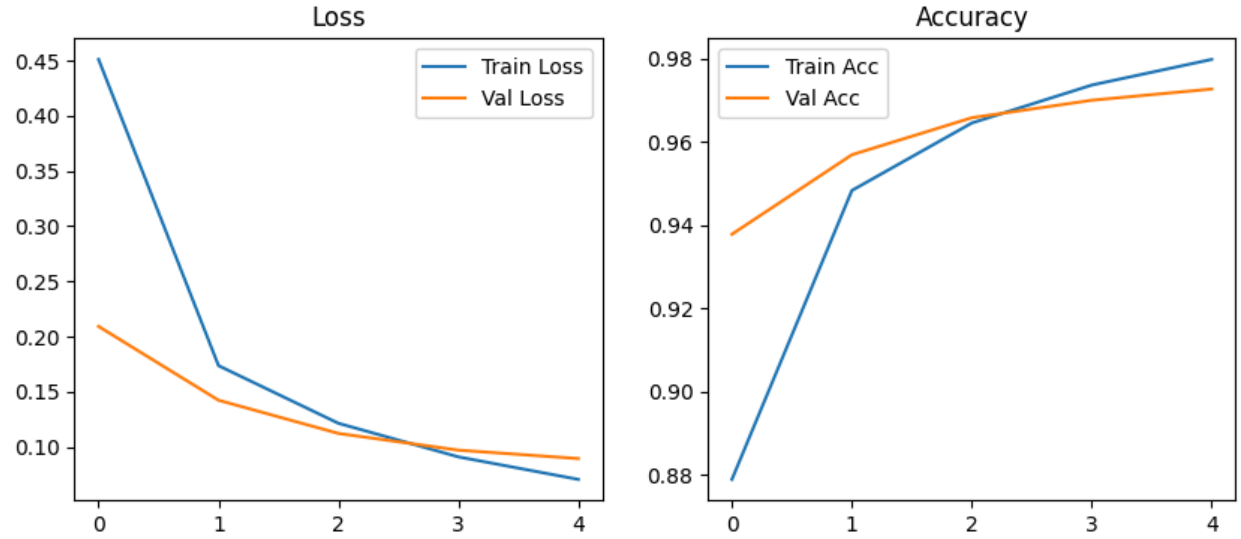
- Implements `ActivationVarianceLogger`, which records variance in activations over epochs.
- Used to compare activation variance between models with and without dropout.

```
#%% [code]
class ActivationVarianceLogger(tf.keras.callbacks.Callback):
    """
    Custom callback to compute and log the variance of activations for Dense layers at th
    """
    def __init__(self, validation_data):
        super(ActivationVarianceLogger, self).__init__()
        self.validation_data = validation_data  # Store validation data for logging

    def set_model(self, model):
        # This method is called by the training loop; here, we build our intermediate mod
        super().set_model(model)
        # Identify Dense layers in the model
        self.dense_layer_indices = [i for i, layer in enumerate(model.layers) if isinstan
        # Collect the outputs of these Dense layers
        layer_outputs = [model.layers[i].output for i in self.dense_layer_indices]
        # Build an intermediate model that maps the input to the outputs of the Dense lay
        self.intermediate_model = tf.keras.Model(inputs=model.inputs[0], outputs=layer_ou

    def on_epoch_end(self, epoch, logs=None):
        # Take a small batch from the validation data (e.g., first 32 samples)
        x_val, _ = self.validation_data[0][:32], self.validation_data[1][:32]
        activations = self.intermediate_model.predict(x_val)
        # Log the variance of activations for each Dense layer to wandb
        for i, act in enumerate(activations):
            variance = np.var(act)
            wandb.log({f"activation_variance_layer_{i+1}": variance}, step=epoch)



#%% [code]
def train_and_visualize_with_variance(config):
```

```
    """
    Trains the model and logs activation variances per hidden layer using the ActivationV
    """
    wandb.init(project="dropout-experiment", config=config)

    if config.get('weight_decay', 0.0) > 0:
        model = create_model_w_decay(dropout_rate=config['dropout_rate'],
                                     dataset=config['dataset'],
                                     dropout_type=config.get('dropout_type', 'standard'),
                                     num_layers=config.get('num_layers', 2),
                                     units=config.get('units', 100),
                                     activation=config.get('activation', 'sigmoid'),
                                     weight_decay=config.get('weight_decay', 0.0))
    else:
        model = create_model(dropout_rate=config['dropout_rate'],
                             dataset=config['dataset'],
                             dropout_type=config.get('dropout_type', 'standard'),
                             num_layers=config.get('num_layers', 2),
                             units=config.get('units', 100),
                             activation=config.get('activation', 'sigmoid'))

    initialize_weights(model, strategy=config.get('weight_init_strategy', 'random'),
                       pretraining_path=config.get('pretraining_path'),
                       threshold=config.get('weight_threshold'))

    optimizer_type = config.get('optimizer_type', 'adam')
    learning_rate = config.get('learning_rate', 0.001)
    if optimizer_type == 'sgd':
        momentum = config.get('momentum', 0.0)
        optimizer = tf.keras.optimizers.SGD(learning_rate=learning_rate, momentum=momentu
    else:
        optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)

    model.compile(optimizer=optimizer,
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    if config['dataset'] == "mnist":
        (x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
        x_train, x_test = x_train / 255.0, x_test / 255.0
    elif config['dataset'] == "cifar10":
        (x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()
        x_train, x_test = x_train / 255.0, x_test / 255.0
    elif config['dataset'] == "reuters":
        (x_train, y_train), (x_test, y_test) = tf.keras.datasets.reuters.load_data(num_wo
        max_length = 100
        from tensorflow.keras.preprocessing.sequence import pad_sequences
        x_train = pad_sequences(x_train, maxlen=max_length)
        x_test = pad_sequences(x_test, maxlen=max_length)
    else:
        raise ValueError("Invalid dataset name.")

    # Create the callback by passing only validation data.
    variance_logger = ActivationVarianceLogger(validation_data=(x_test, y_test))
```

```
        history = model.fit(x_train, y_train, epochs=config['epochs'],
                            validation_data=(x_test, y_test),
                            callbacks=[variance_logger])

        for metric in history.history:
            for epoch, value in enumerate(history.history[metric]):
                wandb.log({metric: value}, step=epoch)

        plt.figure(figsize=(10, 4))
        plt.subplot(1, 2, 1)
        plt.plot(history.history['loss'], label='Train Loss')
        plt.plot(history.history['val_loss'], label='Val Loss')
        plt.title("Loss")
        plt.legend()

        plt.subplot(1, 2, 2)
        plt.plot(history.history['accuracy'], label='Train Acc')
        plt.plot(history.history['val_accuracy'], label='Val Acc')
        plt.title("Accuracy")
        plt.legend()
        plt.show()

        wandb.finish()

    # Example activation variance experiments
    configs_variance = [
        {
            "experiment_name": "NoDropout_Variance",
            "dropout_rate": 0.0,
            "dataset": "mnist",
            "epochs": 5,
            "dropout_type": "standard",
            "num_layers": 2,
            "units": 100,
            "activation": "sigmoid"
        },
        {
            "experiment_name": "Dropout_Variance",
            "dropout_rate": 0.5,
            "dataset": "mnist",
            "epochs": 5,
            "dropout_type": "standard",
            "num_layers": 2,
            "units": 100,
            "activation": "sigmoid"
        }
    ]

    for cfg in configs_variance:
        print("\nRunning Activation Variance Experiment:", cfg["experiment_name"])
        train_and_visualize_with_variance(cfg)
```

```
Running Activation Variance Experiment: NoDropout_Variance
Tracking run with wandb version 0.19.6
Run data is saved locally in /content/wandb/run-20250222_102229-butxxo2f
Syncing run bright-grass-6 to Weights & Biases (docs)
View project at https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment
View run at https://wandb.ai/24mcs110-national-institute-of-technology-hamirpur/dropout-experiment/runs/butxxo2f
Using random weight initialization.
Epoch 1/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 105ms/step
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 5ms/step - accuracy: 0.7793 - loss: 0.8294 - va
Epoch 2/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.9438 - loss: 0.1921 - val
Epoch 3/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 5ms/step - accuracy: 0.9633 - loss: 0.1298 - va
Epoch 4/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.9723 - loss: 0.0957 - va
Epoch 5/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 12s 6ms/step - accuracy: 0.9793 - loss: 0.0734 - va
```



Run history:

accuracy	▁
activation_variance_layer_1	▁▆▇
activation_variance_layer_2	▇▇▅▁
activation_variance_layer_3	▁▆▇
loss	▁
val_accuracy	▁
val_loss	▁

Run summary: