Regularization techniques in Deep Learning:

1. L1 and L2 Regularization
2. Dropout
3. Data Augmentation
4. Early stopping
5. K-Fold Validation
6. Elastic net(Combinnation of L1 and L2 Regularization)

```
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import numpy as np
```

1. L1 Regularization

```
import tensorflow as tf
from tensorflow.keras import layers, models, regularizers
from tensorflow.keras.datasets import fashion_mnist

def create_model_l1(l1_strength=0.01):
    model = models.Sequential()
    model.add(layers.Flatten(input_shape=(28, 28)))  # Flatten the 28x28 images
    model.add(layers.Dense(64, activation='relu', kernel_regularizer=regularizers.l1(l1_strength)))
    model.add(layers.Dense(64, activation='relu', kernel_regularizer=regularizers.l1(l1_strength)))
    model.add(layers.Dense(10, activation='softmax'))
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

# Load Fashion MNIST data
(train, train_label), (test, test_label) = fashion_mnist.load_data()

# Normalize the data
train, test = train / 255.0, test / 255.0

# Create model
model = create_model_l1(l1_strength=0.01)

# Train model
model.fit(train, train_label, epochs=10, batch_size=32)

# Evaluate model
test_loss, test_acc = model.evaluate(test, test_label)
print(f'Test accuracy: {test_acc}')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 ━━━━━━━━━━━━━━━━━━━━ 0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.6635 - loss: 4.3629
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.7400 - loss: 1.2441
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7513 - loss: 1.1641
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.7617 - loss: 1.1123
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7646 - loss: 1.0920
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7699 - loss: 1.0650
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.7781 - loss: 1.0411
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7771 - loss: 1.0332
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.7745 - loss: 1.0281
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7763 - loss: 1.0201
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7551 - loss: 1.0357
```

```
Test accuracy: 0.753600001335144
```

2. L2 Regularization

```
import tensorflow as tf
from tensorflow.keras import layers, models, regularizers
from tensorflow.keras.datasets import fashion_mnist

def create_model_l2(l2_strength=0.01):
    model = models.Sequential()
    model.add(layers.Flatten(input_shape=(28, 28)))  # Flatten the 28x28 images
    model.add(layers.Dense(64, activation='relu', kernel_regularizer=regularizers.l2(l2_strength)))
    model.add(layers.Dense(64, activation='relu', kernel_regularizer=regularizers.l2(l2_strength)))
    model.add(layers.Dense(10, activation='softmax'))
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

# Load Fashion MNIST data
(train, train_label), (test, test_label) = fashion_mnist.load_data()

# Normalize the data
train, test = train / 255.0, test / 255.0

# Create model
model = create_model_l2(l2_strength=0.01)

# Train model
model.fit(train, train_label, epochs=10, batch_size=32)

# Evaluate model
test_loss, test_acc = model.evaluate(test, test_label)
print(f'Test accuracy: {test_acc}')
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.7429 - loss: 1.3746
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8093 - loss: 0.7179
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8221 - loss: 0.6512
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8297 - loss: 0.6181
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8338 - loss: 0.5994
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.8349 - loss: 0.5847
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step - accuracy: 0.8379 - loss: 0.5689
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8393 - loss: 0.5566
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8384 - loss: 0.5552
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8408 - loss: 0.5460
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8452 - loss: 0.5468
Test accuracy: 0.8400999903678894
```

3. Dropout Regularization

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import fashion_mnist

def create_model_dropout(dropout_rate=0.5):
    model = models.Sequential()
    model.add(layers.Flatten(input_shape=(28, 28)))  # Flatten the 28x28 images
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dropout(dropout_rate))
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dropout(dropout_rate))
    model.add(layers.Dense(10, activation='softmax'))
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

# Load Fashion MNIST data
(train, train_label), (test, test_label) = fashion_mnist.load_data()

# Normalize the data
train, test = train / 255.0, test / 255.0
```

```
# Create model
model = create_model_dropout(dropout_rate=0.5)

# Train model
model.fit(train, train_label, epochs=10, batch_size=32)

# Evaluate model
test_loss, test_acc = model.evaluate(test, test_label)
print(f'Test accuracy: {test_acc}')
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.5702 - loss: 1.1773
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.7642 - loss: 0.6540
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.7910 - loss: 0.5974
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.7975 - loss: 0.5713
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8104 - loss: 0.5496
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8095 - loss: 0.5341
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8128 - loss: 0.5261
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.8181 - loss: 0.5181
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8167 - loss: 0.5137
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8214 - loss: 0.5021
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8628 - loss: 0.4011
Test accuracy: 0.8531000018119812
```

4. Early Stopping Regularization

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.datasets import fashion_mnist

def create_model_early_stopping():
    model = models.Sequential()
    model.add(layers.Flatten(input_shape=(28, 28)))  # Flatten the 28x28 images
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(10, activation='softmax'))
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

# Load Fashion MNIST data
(train, train_label), (test, test_label) = fashion_mnist.load_data()

# Normalize the data
train, test = train / 255.0, test / 255.0

# Create model
model = create_model_early_stopping()

# Implement Early Stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)

# Train model with early stopping
model.fit(train, train_label, epochs=10, batch_size=32, validation_split=0.2, callbacks=[early_stopping])

# Evaluate model
test_loss, test_acc = model.evaluate(test, test_label)
print(f'Test accuracy: {test_acc}')
```

```
Epoch 1/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7468 - loss: 0.7198 - val_accuracy: 0.8393 - val_loss: 0.4429
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.8598 - loss: 0.3979 - val_accuracy: 0.8629 - val_loss: 0.3765
Epoch 3/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 8s 3ms/step - accuracy: 0.8661 - loss: 0.3603 - val_accuracy: 0.8578 - val_loss: 0.3787
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.8805 - loss: 0.3270 - val_accuracy: 0.8686 - val_loss: 0.3757
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.8855 - loss: 0.3143 - val_accuracy: 0.8695 - val_loss: 0.3781
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.8902 - loss: 0.3021 - val_accuracy: 0.8738 - val_loss: 0.3572
```

```
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.8930 - loss: 0.2830 - val_accuracy: 0.8836 - val_loss: 0.3269
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8962 - loss: 0.2712 - val_accuracy: 0.8798 - val_loss: 0.3380
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.9023 - loss: 0.2619 - val_accuracy: 0.8801 - val_loss: 0.3309
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9051 - loss: 0.2548 - val_accuracy: 0.8863 - val_loss: 0.3280
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8720 - loss: 0.3584
Test accuracy: 0.8729000091552734
```

5. K-Fold Cross-Validation

```
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.model_selection import KFold
import numpy as np
from tensorflow.keras.datasets import fashion_mnist

def create_model_kfold():
    model = models.Sequential()
    model.add(layers.Flatten(input_shape=(28, 28)))  # Flatten the 28x28 images
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(10, activation='softmax'))
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

# Load Fashion MNIST data
(train, train_label), (test, test_label) = fashion_mnist.load_data()

# Normalize the data
train, test = train / 255.0, test / 255.0

# Reshape for KFold: Flatten the images from (28, 28) to (784)
train = train.reshape(-1, 28, 28)  # Ensure data remains in (28, 28) format
test = test.reshape(-1, 28, 28)

kf = KFold(n_splits=5)
acc_scores = []

for train_index, val_index in kf.split(train):
    # Get the training and validation data
    X_train, X_val = train[train_index], train[val_index]
    y_train, y_val = train_label[train_index], train_label[val_index]

    model = create_model_kfold()
    # Train the model
    model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=0)
    # Evaluate on validation data
    val_loss, val_acc = model.evaluate(X_val, y_val, verbose=0)
    acc_scores.append(val_acc)

# Print the average accuracy across the folds
print(f'Average K-Fold Accuracy: {np.mean(acc_scores)}')
```

```
Average K-Fold Accuracy: 0.8838666558265686
```

6. Elastic Net Regularization

```
import tensorflow as tf
from tensorflow.keras import layers, models, regularizers
from tensorflow.keras.datasets import fashion_mnist

def create_model_elasticnet(l1_strength=0.01, l2_strength=0.01):
    model = models.Sequential()
    model.add(layers.Flatten(input_shape=(28, 28)))  # Flatten the 28x28 images
    model.add(layers.Dense(64, activation='relu',
                           kernel_regularizer=regularizers.l1_l2(l1=l1_strength, l2=l2_strength)))
    model.add(layers.Dense(64, activation='relu', kernel_regularizer=regularizers.l1_l2(l1=l1_strength, l2=l2_strength)))
    model.add(layers.Dense(10, activation='softmax'))
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

# Load Fashion MNIST data
(train, train_label), (test, test_label) = fashion_mnist.load_data()
```

```
# Normalize the data
train, test = train / 255.0, test / 255.0

# Create model
model = create_model_elasticnet(l1_strength=0.01, l2_strength=0.01)

# Train model
model.fit(train, train_label, epochs=10, batch_size=32)

# Evaluate model
test_loss, test_acc = model.evaluate(test, test_label)
print(f'Test accuracy: {test_acc}')
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.6226 - loss: 4.5079
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7372 - loss: 1.2280
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7470 - loss: 1.1530
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.7520 - loss: 1.1166
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.7565 - loss: 1.0888
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.7622 - loss: 1.0655
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7693 - loss: 1.0405
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7663 - loss: 1.0315
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.7689 - loss: 1.0197
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7664 - loss: 1.0166
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7727 - loss: 1.0075
Test accuracy: 0.7724000215530396
```