Deep Neural Networks

Lab 4

Timo Wang

Data Preprocessing

Normalization

- Normalization is a preprocessing step that rescales the input features to a standard range, typically [0, 1] or [-1, 1], ensuring that each feature contributes equally to the model's learning process.
- Two common normalization techniques are **Min-Max Scaling** and **Standard Scaling** (the one you have seen in the lecture).
  - x_scaled = (x - min) / (max - min)
  - x_scaled = (x - mean) / std

```
from sklearn.preprocessing import MinMaxScaler,
StandardScaler

# Min-Max Scaling minmax_scaler = MinMaxScaler()
X_train_minmax = minmax_scaler.fit_transform(X_train)
X_test_minmax = minmax_scaler.transform(X_test)

# Standard Scaling standard_scaler = StandardScaler()
X_train_standard = standard_scaler.fit_transform(X_train)
X_test_standard = standard_scaler.transform(X_test)
```

PCA, Whitening

- Principal Component Analysis (PCA) is a dimensionality reduction technique that projects the original data onto a lower-dimensional subspace, preserving as much variance as possible.
- PCA can be used for data preprocessing to remove correlations between features, reduce overfitting, and improve the model's training speed.
- Whitening, also known as ZCA (Zero Component Analysis) or sphering, is an additional step performed after PCA that scales the principal components to have unit variance, making the data have zero mean and identity covariance matrix.

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler() X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
pca = PCA(n_components=2, whiten=True)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
```

Variance and PCA

- Variance is a measure of the spread of data points in a dataset that provides important insights into the complexity and information content of the data.
- PCA aims at reducing the dimension of the data points while maintaining most of the variance in the data.
- Choosing an optimal number of PCA components is important as too few may lead to underfitting while too many may lead to overfitting
- We can use **explained variance ratio** to quantify the amount of variance captured by each principal component.

Model Design

Weight Initialization

- Glorot (Xavier) weight initialization is a technique used to set the initial weights of a neural network in a way that helps the network learn faster and more effectively.
- The Glorot initialization sets the initial weights by drawing random values from a uniform distribution, with a range that depends on the number of input units in the weight matrix.
- Using Glorot initialization helps prevent vanishing or exploding gradients, which can occur when weights are initialized with inappropriate values, making it challenging for the network to learn.

```
model = Sequential([

    Dense(128, activation='relu‘,
kernel_initializer='glorot_uniform‘, input_shape=(28 *
28,)),

    Dense(64, activation='relu‘,
kernel_initializer='glorot_uniform‘),

    Dense(10, activation='softmax',
kernel_initializer='glorot_uniform‘)

])
```

Regularization and Dropout

- Regularization is a technique used in deep learning to prevent overfitting and to encourage the model to learn simpler representations of the data.
- Common regularization methods include L1 and L2 regularization, which penalize the absolute or squared values of the weights, respectively, and Dropout, which randomly sets a fraction of the input units to zero during training.
- Regularization can help improve generalization performance, making the model more robust to variations in the input data and less likely to fit the noise in the training set.

```
model = Sequential([
    Dense(128, activation='relu‘, kernel_regularizer=l1_l2(l1=1e-5, l2=1e-4), input_shape=(28 * 28,)),
    Dropout(0.5),
    Dense(64, activation='relu', kernel_regularizer=l1_l2(l1=1e-5, l2=1e-4)),
    Dropout(0.5),
    Dense(10, activation='softmax‘)
])
```

Batch Normalization

- Batch normalization is a technique used in deep learning to improve the training process by normalizing the activations of each layer, making the input distribution more stable across different layers.
- Batch normalization computes the mean and variance of each feature within a mini-batch and normalizes the activations accordingly, followed by a learnable scale and shift parameter.

```
model = Sequential([
    Dense(128, activation='relu', input_shape=(28 * 28,)),
    BatchNormalization(),
    Dense(64, activation='relu‘),
    BatchNormalization(),
    Dense(10, activation='softmax‘)
])
```

Hands On

Data Preprocessing and Regularization Techniques

1. Import all necessary packages
2. Load and preprocess the data
3. Define a custom Tensorboard callback for saving training results
4. Define a function to create a neural network model with the following architecture, accepting hyperparameters as input arguments. You can also try different architecture too:
    1. Input layer with the appropriate input shape
    2. Dense layer with a variable number of units, ReLU activation, Glorot weight initialization, and L1/L2 regularization
    3. Batch Normalization layer
    4. Dropout layer with a variable dropout rate
    5. Dense layer with a variable number of units, ReLU activation, Glorot weight initialization, and L1/L2 regularization
    6. Batch Normalization layer
    7. Dropout layer with a variable dropout rate
    8. Output layer with the appropriate number of units and softmax activation
5. Define the parameters you want to tune
6. Setup logging with Tensorboard and train for each set of parameters
7. Create a new model with the best-found parameter set and evaluate the model