VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**Nguyễn Đình Việt Hoàng – 522H0120**

**Nguyễn Minh Quân – 521H0473**

**FINAL REPORT**

**MACHINE LEARNING**

**HO CHI MINH CITY, YEAR 2024**

VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**Nguyễn Đình Việt Hoàng – 522H0120**

**Nguyễn Minh Quân – 521H0473**

**FINAL REPORT**

**MACHINE LEARNING**

**Instructor**

**Mr. Lê Anh Cường**

**HO CHI MINH CITY, YEAR 2024**

**ACKNOWLEDGEMENT**

We sincerely thank Professor Lê Anh Cường for teaching us the Software Engineering course with great enthusiasm. We want to express our deep appreciation for the dedication and professional knowledge that you shared with us. Through your classes, we gained a better understanding of the fundamental aspects of the Software Engineering, thanks to your detailed explanations and practical applications. You helped us grasp the knowledge and apply it effectively. Finally, we extend our heartfelt gratitude to Professor Lê Anh Cường for your commitment and invaluable support throughout our learning journey in this course. The skills and knowledge we acquired will continue to impact our future development. We sincerely thank you and wish your health, success, and happiness.

*Ho Chi Minh City, May 18, 2024*

*Authors:*

*Nguyễn Đình Việt Hoàng*

*Nguyễn Minh Quân*

**THE COMPLETION REPORT HAS BEEN SUBMITTED AT TON DUC THANG UNIVERSITY**

Our group assures that this is our own report and was guided by Professor Lê Anh Cường. The research content and results in this report are honest and have not been published in any form before. The figures in the tables used for analysis, comments, and evaluations were collected by the authors from various sources clearly stated in the reference section.

Additionally, the report includes some comments, evaluations, and data from other authors and organizations, all of which are cited and noted for their origin.

**If any fraud is detected, we fully take responsibility for the content of our final report for the second semester of the 2023-2024 academic year.** Ton Duc Thang University is not involved in any copyright or intellectual property violations that we may cause during the process (if any).

*Ho Chi Minh City, May 18, 2024*

*Authors:*

*Nguyễn Đình Việt Hoàng*

*Nguyễn Minh Quân*

**INSTRUCTOR RUBRIC**

Supervisor’s Name: ……………………………………………………………..............

Comments: ……………………………………………………………………………...

Total Score Based on Rubric Evaluation: ……………………………………………….

*Ho Chi Minh City, date … month … year …*

*Supervisor*

*(sign and write your full name)*

**TABLE OF CONTENTS**

[QUESTION 1 7](#_Toc166936726)

[a) Discuss different optimizers used in training Neural Networks. Describe the advantages and disadvantages of these methods. 7](#_Toc166936727)

[1. Stochastic Gradient Descent (SGD) 7](#_Toc166936728)

[2. Mini-Batch Gradient Descent 7](#_Toc166936729)

[3. Momentum 8](#_Toc166936730)

[4. Nesterov Accelerated Gradient (NAG) 8](#_Toc166936731)

[5. Adagrad (Adaptive Gradient Algorithm) 8](#_Toc166936732)

[6. RMSprop (Root Mean Square Propagation) 8](#_Toc166936733)

[7. Adam (Adaptive Moment Estimation) 9](#_Toc166936734)

[8. Nadam (Nesterov-accelerated Adaptive Moment Estimation) 9](#_Toc166936735)

[9. AdaMax 9](#_Toc166936736)

[10. AMSGrad 9](#_Toc166936737)

[Compare these methods for the same problem (using the same dataset). 10](#_Toc166936738)

[b) Explain the backpropagation algorithm for learning parameters in Neural Networks and provide illustrative code. 14](#_Toc166936739)

[1. Forward Pass 14](#_Toc166936740)

[2. Loss Calculation 14](#_Toc166936741)

[3. Backward Pass 14](#_Toc166936742)

[4. Parameter Update 15](#_Toc166936743)

[Example 15](#_Toc166936744)

[Conclusion 16](#_Toc166936745)

[QUESTION 2a 16](#_Toc166936746)

[a) Discuss the Convolutional Neural Networks (CNN) model and its application to image classification problems. 16](#_Toc166936747)

[Compare it with the traditional Feedforward Neural Network (FFNN) model. 19](#_Toc166936748)

[Conclusion 19](#_Toc166936749)

[b) Use different methods to handle overfitting for the above models and compare the results. 20](#_Toc166936750)

[c) Research methods to improve the model, for example, through feature selection or hyperparameter optimization. 20](#_Toc166936751)

# QUESTION 1

## a) Discuss different optimizers used in training Neural Networks. Describe the advantages and disadvantages of these methods.

### 1. Stochastic Gradient Descent (SGD)

SGD updates the weights of the network incrementally for each training sample.

Advantages:

* Simplicity: Easy to understand and implement.
* Efficiency: Suitable for large datasets as it processes one sample at a time.

Disadvantages:

* Convergence: May have slow convergence and get stuck in local minima.
* Noise: High variance in updates can lead to noisy convergence.

### 2. Mini-Batch Gradient Descent

Mini-Batch Gradient Descent is a compromise between SGD and Batch Gradient Descent. It updates the weights based on a small random subset of the data (mini-batch).

Advantages:

* Speed: Faster convergence than Batch Gradient Descent.
* Stability: Reduces the variance in updates compared to SGD.

Disadvantages:

* Complexity: Requires careful tuning of the mini-batch size.
* Memory: Requires more memory than SGD.

### 3. Momentum

Momentum accelerates SGD by adding a fraction of the previous update vector to the current update vector.

Advantages:

* Speed: Faster convergence and helps escape local minima.
* Stability: Smoother updates compared to vanilla SGD.

Disadvantages:

* Parameters: Requires tuning of the momentum parameter.

### 4. Nesterov Accelerated Gradient (NAG)

NAG is an extension of momentum that anticipates the future position of the parameters and corrects them before computing the gradient.

Advantages:

* Speed: Faster convergence than standard momentum.
* Precision: More accurate updates.

Disadvantages:

* Complexity: More complex to implement and requires additional parameter tuning.

### 5. Adagrad (Adaptive Gradient Algorithm)

Adagrad adapts the learning rate for each parameter based on the historical gradients.

Advantages:

* Adaptability: Suitable for sparse data and handles varying learning rates automatically.
* Parameter-Free: Reduces the need to manually tune the learning rate.

Disadvantages:

* Convergence: Learning rate can become very small, causing the training to stop prematurely.

### 6. RMSprop (Root Mean Square Propagation)

RMSprop is a modification of Adagrad that deals with the diminishing learning rate problem by using a moving average of squared gradients.

Advantages:

* Stability: Maintains a good learning rate throughout the training.
* Convergence: Faster and more reliable convergence compared to Adagrad.

Disadvantages:

* Parameters: Requires tuning of the decay parameter.

### 7. Adam (Adaptive Moment Estimation)

Adam combines the advantages of both RMSprop and momentum by maintaining a moving average of both the gradients and the squared gradients.

Advantages:

* Efficiency: Suitable for large datasets and high-dimensional parameter spaces.
* Adaptability: Automatically adjusts the learning rate.
* Convergence: Generally fast and reliable convergence.

Disadvantages:

* Parameters: Requires tuning of several hyperparameters (learning rate, beta1, beta2).

### 8. Nadam (Nesterov-accelerated Adaptive Moment Estimation)

Nadam is an extension of Adam that incorporates Nesterov momentum.

Advantages:

* Speed: Faster convergence than Adam in some cases.
* Precision: Combines the benefits of Nesterov momentum and Adam.

Disadvantages:

* Complexity: More complex and requires tuning of additional parameters.

### 9. AdaMax

AdaMax is a variant of Adam that uses the infinity norm instead of the L2 norm.

Advantages:

* Stability: More stable in some cases compared to Adam.
* Robustness: Handles large gradients better.

Disadvantages:

* Specificity: May not always outperform Adam.

### 10. AMSGrad

AMSGrad is a variant of Adam that fixes some issues with the convergence of Adam by using the maximum of past squared gradients.

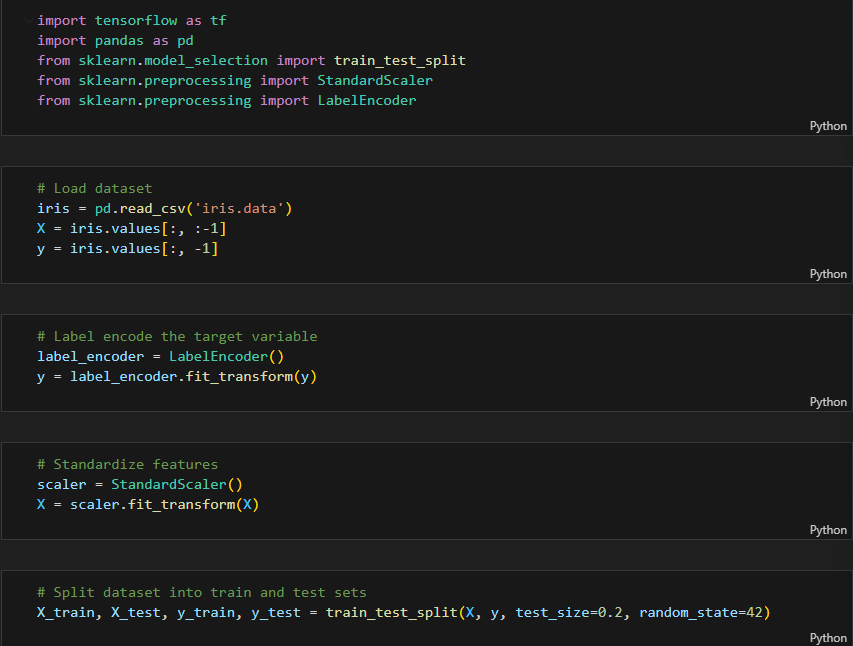
Advantages:

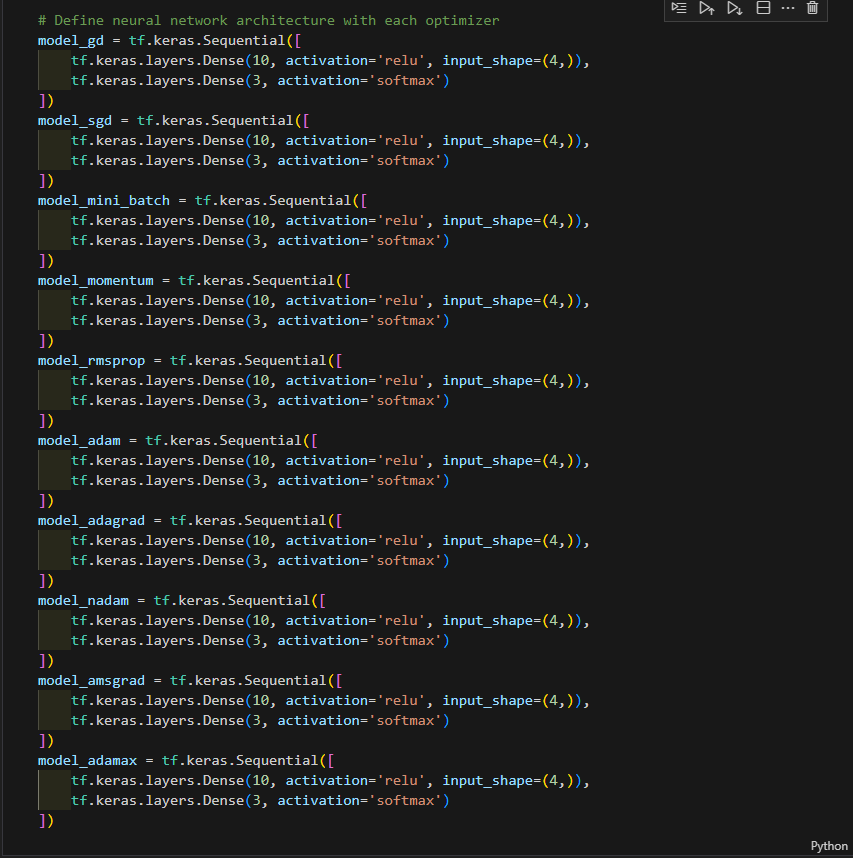
* Convergence: Improved theoretical guarantees of convergence.

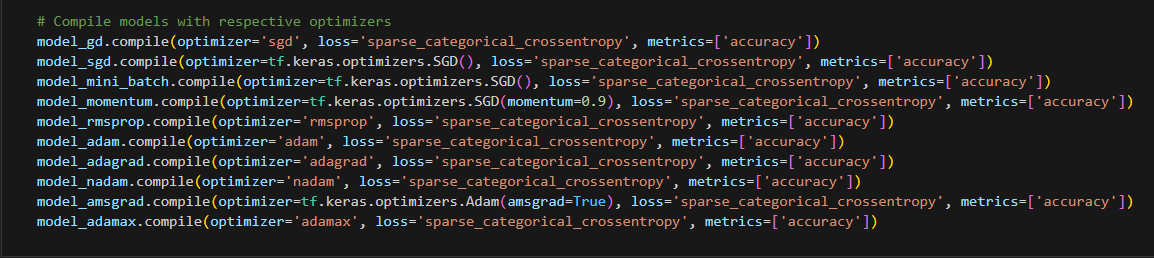
Disadvantages:

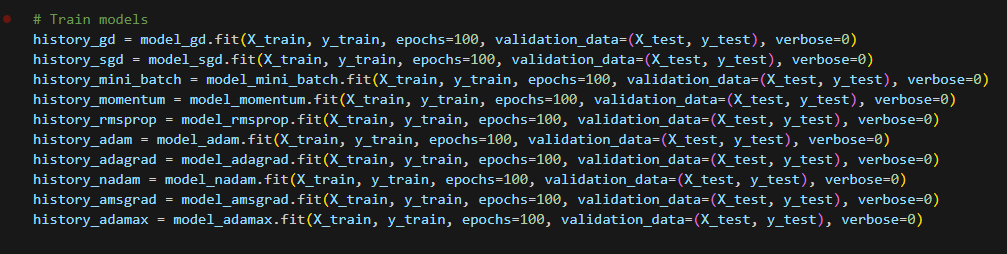
* Complexity: Slightly more complex to implement than Adam.

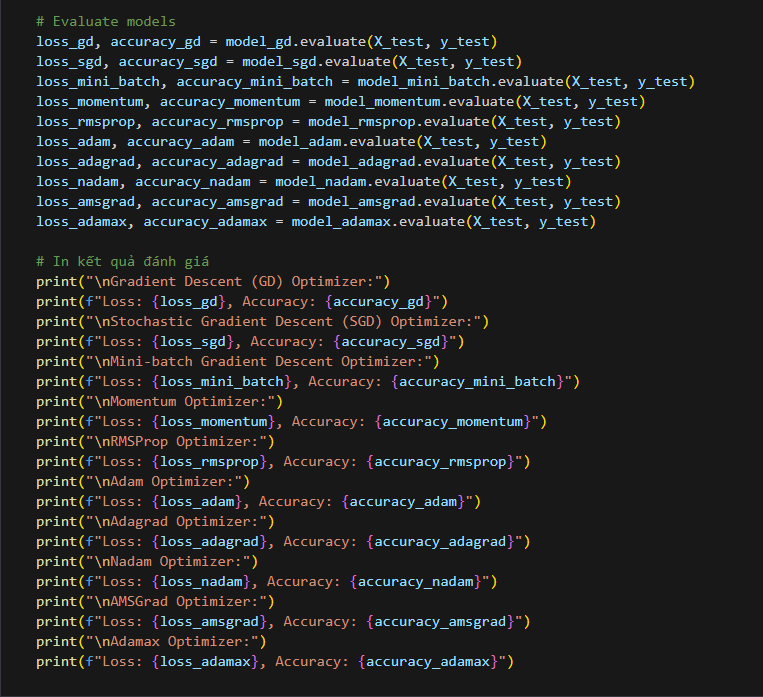
## Compare these methods for the same problem (using the same dataset).

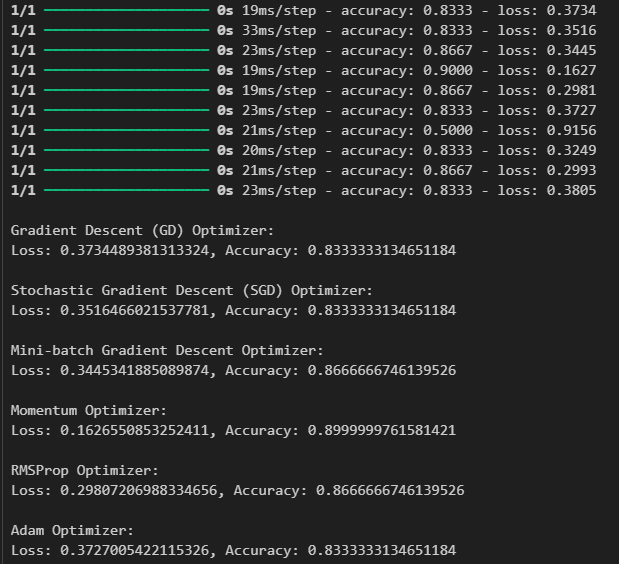


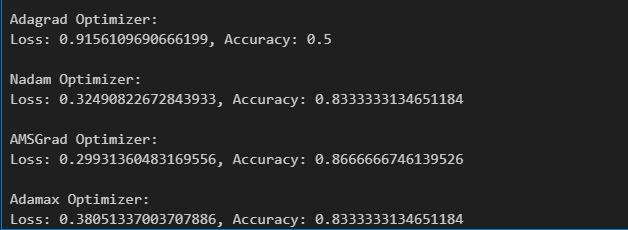












## b) Explain the backpropagation algorithm for learning parameters in Neural Networks and provide illustrative code.

Backpropagation is an algorithm used for training neural networks by adjusting the weights to minimize the error between the predicted output and the actual output. It involves a forward pass, a backward pass, and an update step. Here’s a detailed explanation of the backpropagation algorithm:

### 1. **Forward Pass**

In the forward pass, the input data is passed through the network, layer by layer, to compute the output. For each neuron in the network, the following operations are performed:

* **Linear Combination:** Compute the weighted sum of the inputs plus a bias term.

z=∑iwixi+bz = \sum\_{i} w\_i x\_i + bz=i∑​wi​xi​+b

where wiw\_iwi​ are the weights, xix\_ixi​ are the inputs, and bbb is the bias.

* **Activation Function:** Apply a nonlinear activation function to the linear combination to get the output of the neuron.

a=f(z)a = f(z)a=f(z)

where fff is the activation function.

### 2. **Loss Calculation**

The output from the forward pass is compared to the actual target values to compute the loss. A common loss function for regression is Mean Squared Error (MSE), and for classification, it might be Cross-Entropy Loss.

### 3. **Backward Pass**

In the backward pass, the error is propagated backward through the network to compute the gradient of the loss function with respect to each weight. This involves two main steps:

#### 3.1 **Output Layer Gradients**

For the output layer, the gradient of the loss with respect to the output of the neuron is computed first. Let LLL be the loss function and aLa^LaL be the output of the final layer:

∂L∂aL=gradient of loss with respect to output\frac{\partial L}{\partial a^L} = \text{gradient of loss with respect to output}∂aL∂L​=gradient of loss with respect to output

#### 3.2 **Hidden Layer Gradients**

For hidden layers, the gradients are computed using the chain rule. For a neuron in layer lll, the gradient of the loss with respect to the weights and biases is calculated as follows:

* **Gradient with respect to weights:**

∂L∂wl=δl⋅(al−1)T\frac{\partial L}{\partial w^l} = \delta^l \cdot (a^{l-1})^T∂wl∂L​=δl⋅(al−1)T

where δl=∂L∂zl=∂L∂al⋅f′(zl)\delta^l = \frac{\partial L}{\partial z^l} = \frac{\partial L}{\partial a^l} \cdot f'(z^l)δl=∂zl∂L​=∂al∂L​⋅f′(zl) and al−1a^{l-1}al−1 is the activation from the previous layer.

* **Gradient with respect to biases:**

∂L∂bl=δl\frac{\partial L}{\partial b^l} = \delta^l∂bl∂L​=δl

The term δl\delta^lδl is the error term for layer lll.

### 4. **Parameter Update**

Once the gradients are computed for each weight and bias, the parameters are updated using an optimization algorithm, commonly gradient descent:

wl:=wl−η∂L∂wlw^l := w^l - \eta \frac{\partial L}{\partial w^l}wl:=wl−η∂wl∂L​

bl:=bl−η∂L∂blb^l := b^l - \eta \frac{\partial L}{\partial b^l}bl:=bl−η∂bl∂L​

where η\etaη is the learning rate.

### Example

Consider a simple neural network with one hidden layer:

1. **Forward Pass:**
   * Compute activations for the hidden layer: a1=f(W1x+b1)a^1 = f(W^1 x + b^1)a1=f(W1x+b1)
   * Compute output layer: a2=f(W2a1+b2)a^2 = f(W^2 a^1 + b^2)a2=f(W2a1+b2)
2. **Loss Calculation:**
   * Compute the loss L(y,a2)L(y, a^2)L(y,a2)
3. **Backward Pass:**
   * Compute gradient of loss with respect to output: ∂L∂a2\frac{\partial L}{\partial a^2}∂a2∂L​
   * Compute δ2=∂L∂a2⋅f′(z2)\delta^2 = \frac{\partial L}{\partial a^2} \cdot f'(z^2)δ2=∂a2∂L​⋅f′(z2)
   * Compute ∂L∂W2=δ2⋅(a1)T\frac{\partial L}{\partial W^2} = \delta^2 \cdot (a^1)^T∂W2∂L​=δ2⋅(a1)T
   * Compute ∂L∂b2=δ2\frac{\partial L}{\partial b^2} = \delta^2∂b2∂L​=δ2
   * Propagate error back to hidden layer: δ1=(W2)Tδ2⋅f′(z1)\delta^1 = (W^2)^T \delta^2 \cdot f'(z^1)δ1=(W2)Tδ2⋅f′(z1)
   * Compute ∂L∂W1=δ1⋅xT\frac{\partial L}{\partial W^1} = \delta^1 \cdot x^T∂W1∂L​=δ1⋅xT
   * Compute ∂L∂b1=δ1\frac{\partial L}{\partial b^1} = \delta^1∂b1∂L​=δ1
4. **Update Parameters:**
   * Update weights and biases using gradient descent.

### Conclusion

Backpropagation efficiently computes the gradient of the loss function with respect to each parameter in the network, enabling the use of gradient-based optimization algorithms to train deep neural networks. Despite its simplicity, backpropagation is a powerful algorithm that forms the backbone of modern neural network training.

# QUESTION 2a

## a) Discuss the Convolutional Neural Networks (CNN) model and its application to image classification problems.

Convolutional Neural Networks (CNNs) are a class of deep neural networks specifically designed to process and analyze grid-like data structures, such as images. They leverage convolutional layers to automatically learn spatial hierarchies of features from input images.

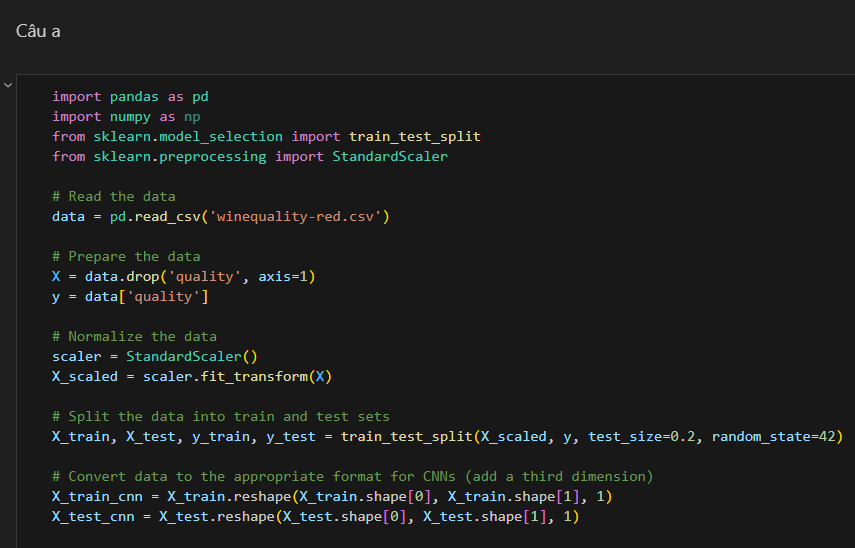
#### Architecture

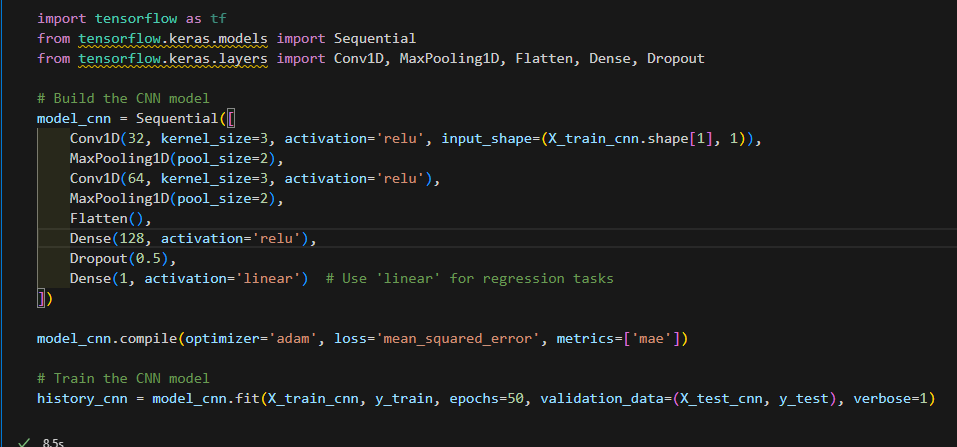
The architecture of CNNs typically consists of several key components:

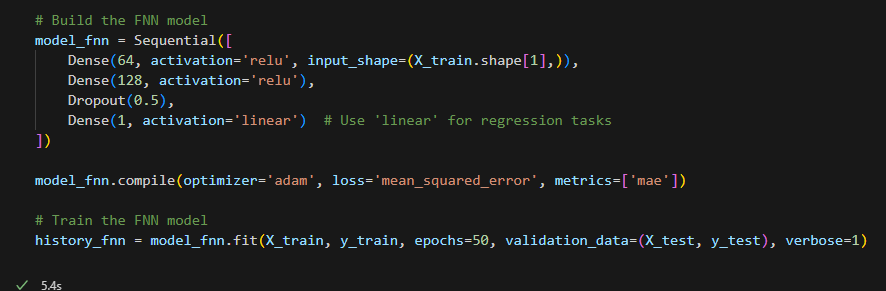
1. **Convolutional Layers:**
   * These layers apply convolutional operations to the input, using a set of filters (kernels) that slide over the input data.
   * The filters help detect local patterns such as edges, textures, or other relevant features.
2. **Activation Layers:**
   * After each convolutional layer, an activation function, usually ReLU (Rectified Linear Unit), is applied to introduce non-linearity into the model.
3. **Pooling Layers:**
   * Pooling layers (e.g., Max Pooling) reduce the dimensionality of the feature maps by down-sampling, retaining the most significant information.
   * This helps in reducing the computational complexity and preventing overfitting.
4. **Fully Connected Layers:**
   * After several convolutional and pooling layers, the high-level reasoning in the network is performed via fully connected layers.
   * These layers resemble the traditional neural networks and are used to make predictions based on the extracted features.
5. **Output Layer:**
   * The final layer of the network typically uses a softmax activation function for classification tasks or a linear activation for regression tasks.

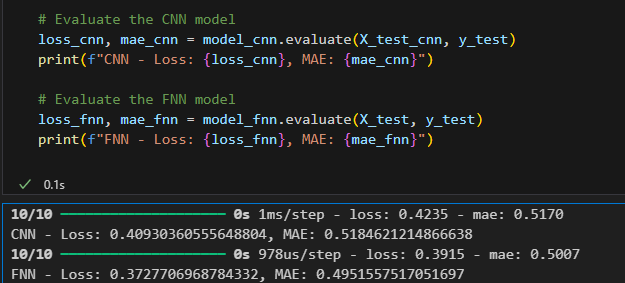
#### Application to Image Classification

CNNs are widely used for image classification due to their ability to automatically and efficiently learn spatial hierarchies of features. Here's a simple example of a CNN applied to image classification:









Feedforward Neural Networks (FFNNs) are the simplest form of artificial neural networks. They consist of an input layer, one or more hidden layers, and an output layer. Each layer is fully connected to the subsequent layer, and information flows in one direction—from input to output.

#### Architecture

The architecture of FFNNs includes:

1. **Input Layer:**
   * Takes the input features and passes them to the first hidden layer.
2. **Hidden Layers:**
   * Consist of neurons that apply a weighted sum of inputs followed by an activation function.
   * Multiple hidden layers can be stacked to create deep networks capable of learning complex representations.
3. **Output Layer:**
   * Produces the final predictions using an activation function like softmax for classification or linear for regression.

#### Limitations in Image Classification

While FFNNs can theoretically handle image data, they are not well-suited for such tasks due to several limitations:

* **High Dimensionality:**
  + Images have high dimensionality, leading to a large number of parameters if processed directly by FFNNs.
* **Spatial Information:**
  + FFNNs do not inherently capture spatial hierarchies and local patterns in images.
* **Computational Complexity:**
  + Directly processing images with FFNNs is computationally expensive and prone to overfitting.

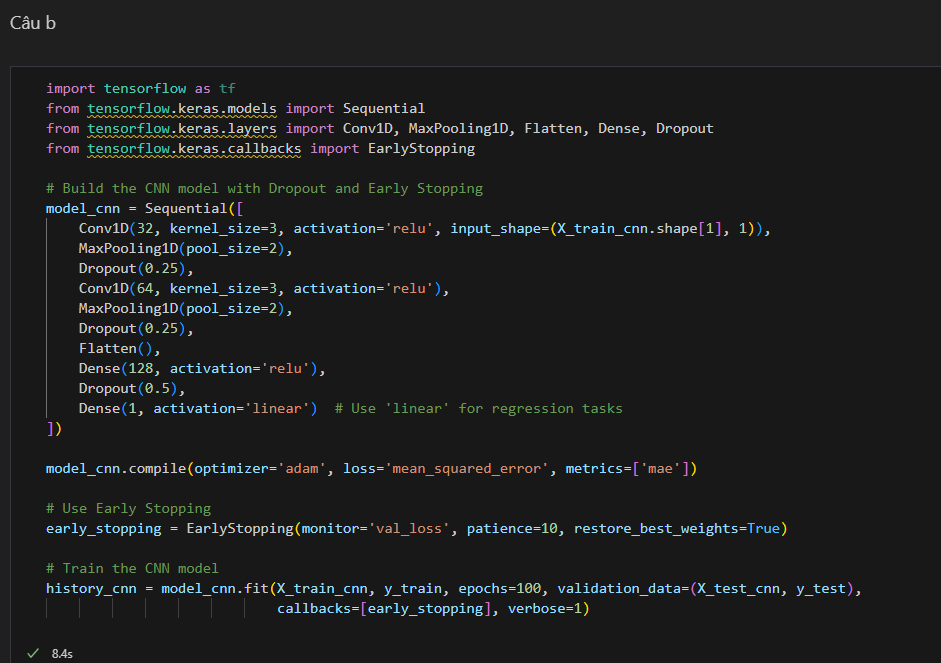
## Compare it with the traditional Feedforward Neural Network (FFNN) model.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | Feature | | |  | | --- | | CNNs | | |  | | --- | | FFNNs | |
| Feature Extraction | Automatically extracts spatial features using convolutional layers. | Requires manual feature extraction or flattening of input data. |
| Handling of Spatial Data | Excels at capturing spatial hierarchies and local patterns. | Lacks mechanisms to capture spatial hierarchies effectively. |
| Number of Parameters | Fewer parameters due to weight sharing in convolutional layers. | More parameters, leading to higher risk of overfitting. |
| Computational Efficiency | More efficient for image data due to pooling and convolutions. | Less efficient and more computationally expensive for images. |
| Performance in Image Tasks | Superior performance in image classification and related tasks. | Inferior performance in image classification tasks. |
| Usage | Widely used in computer vision, image processing, and related fields. | Used in various domains but less effective for image-related tasks. |

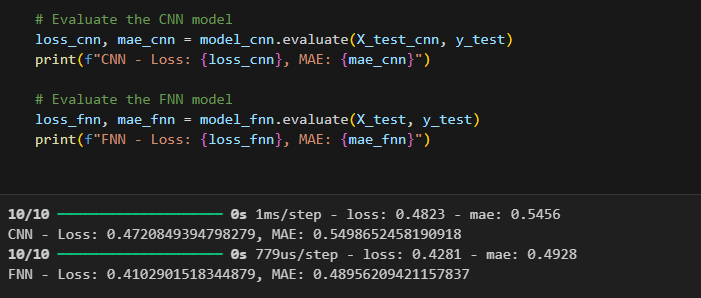
### Conclusion

Convolutional Neural Networks (CNNs) are specifically designed to handle image data effectively by leveraging convolutional layers to automatically learn spatial hierarchies of features. In contrast, Feedforward Neural Networks (FFNNs) lack the ability to inherently capture spatial patterns, making them less suitable for image classification tasks. CNNs outperform FFNNs in both efficiency and accuracy for image-related applications, making them the preferred choice in computer vision and related fields.

## b) Use different methods to handle overfitting for the above models and compare the results.







## c) Research methods to improve the model, for example, through feature selection or hyperparameter optimization.

Improving a machine learning model involves several techniques, such as feature selection and hyperparameter optimization, among others. Here, we'll discuss some of these methods in detail, explaining how they can enhance model performance.

### 1. Feature Selection

Feature selection is the process of selecting a subset of relevant features for use in model construction. This can improve model performance by reducing overfitting, improving accuracy, and decreasing computational cost.

#### Methods for Feature Selection:

**a. Filter Methods:**

* **Correlation Coefficient:**
  + Calculate the correlation between each feature and the target variable.
  + Select features with high correlation (absolute value) to the target.
* **Chi-Square Test:**
  + For categorical features, this test measures the dependency between the feature and the target variable.
* **Mutual Information:**
  + Measures the amount of information obtained about one random variable through another random variable.

**b. Wrapper Methods:**

* **Recursive Feature Elimination (RFE):**
  + Uses the model itself to select features by recursively removing the least important features and building the model until the specified number of features is reached.
* **Forward/Backward Feature Selection:**
  + Iteratively adds (forward) or removes (backward) features based on model performance.

**c. Embedded Methods:**

* **Lasso (L1 Regularization):**
  + Adds a penalty equal to the absolute value of the magnitude of coefficients, driving some coefficients to zero.
* **Tree-Based Methods:**
  + Feature importance scores can be obtained from models like Random Forest, Gradient Boosting, etc.

### 2. Hyperparameter Optimization

Hyperparameter optimization involves finding the optimal set of hyperparameters for a learning algorithm to improve its performance.

#### Methods for Hyperparameter Optimization:

**a. Grid Search:**

* Exhaustively searches through a specified parameter grid.
* Computationally expensive, especially for large grids and datasets.

**b. Random Search:**

* Randomly samples the parameter space.
* Less exhaustive but computationally cheaper than grid search.

**c. Bayesian Optimization:**

* Uses Bayesian techniques to model the performance of the hyperparameters and select the next set of parameters to evaluate.
* More efficient than grid and random search.

**d. Hyperband:**

* An early stopping method that speeds up random search by allocating more resources to promising configurations.

### 3. Data Augmentation

Especially useful in image data to artificially increase the size of the training set by creating modified versions of images in the dataset.

### 4. Model Ensembles

Combining predictions from multiple models to improve performance.

**a. Bagging:**

* Uses models like Random Forest where multiple decision trees are trained on random subsets of data and their predictions are averaged.

**b. Boosting:**

* Models like Gradient Boosting and XGBoost where each new model attempts to correct the errors of the previous model.

**c. Stacking:**

* Combines multiple different models (base models) and a meta-model that learns to combine their predictions.

### Conclusion

Improving machine learning models can be achieved through various methods like feature selection, hyperparameter optimization, data augmentation, and model ensembling. These techniques help in enhancing model performance, reducing overfitting, and making the model more generalizable to unseen data. Experimenting with and combining these methods can lead to significant improvements in model accuracy and robustness.

