IDL Assignment 1

Alexander Scheerder - s3692442
Kenji Opdam - s3683591
Godwin Addetsi - s4251938

***Abstract*— This report presents a comprehensive exploration of neural network architectures applied to image classification tasks using TensorFlow and Keras. The assignment is divided into two main tasks. Task 1 focuses on building and evaluating Multi-Layer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) for classification on the Fashion MNIST and CIFAR-10 datasets. Various hyperparameters such as activation functions, optimizers, regularization techniques, and network depth are systematically tested to understand their impact on model performance. Task 2 involves designing a CNN capable of interpreting analog clock images to predict time, formulated as both classification and regression problems. The goal is to minimize the "common sense" error in time prediction. The experiments provide insights into architectural choices and training strategies that influence accuracy and generalization across tasks.**

I. Introduction

Deep learning has become a cornerstone in modern computer vision tasks, offering powerful tools for image classification, regression, and structured prediction [4]. This report presents two distinct tasks aimed at exploring the capabilities of neural networks using the TensorFlow/Keras framework.

Task 1 investigates the performance of Multi-Layer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) on two benchmark datasets: Fashion MNIST and CIFAR-10. The goal is to understand how architectural choices and hyperparameter tuning affect model accuracy and generalization.

Task 2 focuses on a more complex challenge which is, designing a CNN to interpret analog clock images and predict the time shown. This task is approached through multiple formulations, including classification, regression, and multi-head architectures, with the primary objective of minimizing the "common sense" error in time prediction.

All experiments are conducted using GPU-accelerated environments to ensure efficient training. The following sections detail the methodology, results, and insights gained from each task.

II. Task 1: Background

This section explores the application of neural networks to image classification using two benchmark datasets: Fashion MNIST and CIFAR-10. The Fashion MNIST dataset consists of 70,000 grayscale images of clothing items, each sized 28×28 pixels and categorized into 10 classes. CIFAR-10 contains 60,000 color images sized $32 \times 32 \times 3$, also divided into 10 classes, but with more complex visual features.

Two types of neural network architectures are considered: Multi-Layer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs). A Multi-Layer Perceptron (MLP) is a feedforward neural network composed of fully connected layers, typically used for structured data or flattened image inputs[1]. A Convolutional Neural Network (CNN), on the other hand, is designed to capture spatial hierarchies in image data through convolutional and pooling layers [2].

The experiments aim to evaluate how different architectural choices and hyperparameter configurations affect model performance. These include activation functions, optimizers, regularization techniques, and network depth. The models are trained and validated on Fashion MNIST, and the best-performing configurations are then tested on CIFAR-10 to assess generalization capability.

III. Task 1: Methodology

A. Dataset Preparation

The Fashion MNIST dataset was normalized to the $[0, 1]$ range and split into training, validation, and test sets with a ratio of 80%/10% for training and validation and 10% for test. The CIFAR-10 dataset was similarly preprocessed and split into 80% training, 10% validation, and 10% test sets.

B. Model Architectures

Two primary architectures were implemented:

- **Multi-Layer Perceptron (MLP)**: A feedforward network with fully connected layers. The base model consisted of two hidden layers with 300 and 100 units respectively, using ReLU activation and 10 units softmax output.
- **Convolutional Neural Network (CNN)**: A sequential model with two convolutional layers with base configuration of 32 and 64 filters respectively with ReLU activation, each followed by 2×2 max-pooling, a dense layer of 128 units with dropout of 0.5 regularization, and a softmax output layer of 10 units.

C. Hyperparameter Experiments

A series of controlled experiments were conducted to evaluate the impact of different hyperparameters on validation accuracy:

- **Activation Functions**: ReLU, Tanh, and Sigmoid
- **Optimizers**: Adam, SGD with momentum, and RMSprop
- **Regularization**: Dropout rates of 0.0, 0.3, and 0.5; L2 regularization with $\lambda = 0.001$
- **Network Depth**: MLPs with 1 to 4 hidden layers
- **CNN Filter Configurations**: Three setups with varying filter sizes and dropout rates

D. Transfer to CIFAR-10

The best-performing MLP and CNN configurations from Fashion MNIST were adapted to the CIFAR-10 dataset. The MLP was modified to flatten RGB inputs, while the CNN was extended with additional convolutional layers to better capture spatial features.

E. Training Setup

All models were trained using the sparse categorical cross-entropy loss and accuracy as the evaluation metric. Training was conducted for 10 to 30 epochs depending on the experiment, with a batch size of 128. Validation accuracy was recorded at each epoch to monitor performance.

IV. Task 1: Results

A. Fashion MNIST Performance

The baseline Multi-Layer Perceptron (MLP) achieved a test accuracy of 89.39% after 30 epochs, while the Convolutional Neural Network (CNN) reached 91.74% after 12 epochs. This confirms the superior performance of CNNs for image classification tasks due to their ability to capture spatial hierarchies [3]. Even though it was trained with less number of epochs.

B. Hyperparameter Tuning

Activation Functions: Among ReLU, Tanh, and Sigmoid, Tanh yielded the highest validation accuracy (88.88%), followed closely by ReLU (88.28%).

Optimizers: Adam performed best with a validation accuracy of 88.98%, outperforming RMSprop (88.05%) and SGD (87.50%).

Regularization: A dropout rate of 0.3 without L2 regularization gave the highest validation accuracy (88.38%). Increasing dropout to 0.5 and adding L2 regularization reduced performance.

Network Depth: A depth of 3 hidden layers yielded the best validation accuracy (88.53%), while deeper networks showed diminishing returns.

CNN Configurations: The best CNN setup used 64 and 128 filters with a dropout rate of 0.3, achieving a validation accuracy of 92.47%.

C. CIFAR-10 Transfer Results

The best MLP and CNN models were adapted to the CIFAR-10 dataset and were trained for 15 epochs. The MLP achieved a test accuracy of 43.25%, while the CNN reached 77.24%, demonstrating the CNN's superior generalization to more complex image data. See Figure 1 for visual interpretation of how the two models accuracy increase after each epoch.

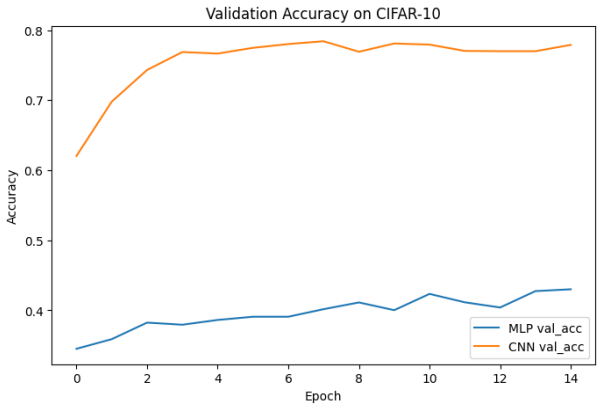

Fig. 1. Validation Accuracy on CIFAR-10 for MLP and CNN

TABLE I
Validation Accuracy Summary for Fashion MNIST and CIFAR-10

Model / Configuration	Fashion MNIST	CIFAR-10
Baseline MLP	89.39%	43.25%
Baseline CNN	91.74%	77.24%
Best Activation (Tanh)	88.88%	–
Best Optimizer (Adam)	88.98%	–
Best Regularization (Dropout 0.3)	88.38%	–
Best Depth (3 layers)	88.53%	–
Best CNN Config (64/128 filters, Dropout 0.3)	92.47%	–

V. Task 1: Discussion

The experiments conducted on Fashion MNIST and CIFAR-10 reveal several important insights into the behavior of neural network architectures and their sensitivity to hyperparameter choices.

The CNN consistently outperformed the MLP across both datasets. On Fashion MNIST, the CNN achieved a test accuracy of 91.74% compared to 89.39% for the MLP. This performance gap widened significantly on CIFAR-10, where the CNN reached 77.24% while the MLP plateaued at 43.25%. These results highlight the importance of spatial feature extraction in image classification tasks, especially for complex datasets.

Activation functions played a subtle but noticeable role, with Tanh slightly outperforming ReLU and Sigmoid. Optimizer choice had a more pronounced effect, with Adam yielding the highest validation accuracy. Regularization experiments showed that moderate dropout (0.3) improved generalization, while excessive dropout combined with L2 regularization degraded performance.

Increasing network depth improved accuracy up to a point, with three hidden layers performing best. However, deeper networks did not yield further gains, suggesting diminishing returns and potential overfitting.

CNN configuration experiments demonstrated that increasing filter sizes and reducing dropout led to better performance. The best configuration used 64 and 128 filters with a dropout rate of 0.3, achieving 92.47% validation accuracy.

Transferring models to CIFAR-10 confirmed that CNNs generalize better to more complex image data. The performance gap between MLP and CNN was visually evident in the validation accuracy plot, Figure 1, reinforcing the architectural advantages of convolutional layers.

Overall, Task 1 provided valuable practical experience in tuning neural networks and understanding the trade-offs involved in architecture and hyperparameter selection.

VI. Task 1: Conclusion

The experiments conducted in Task 1 demonstrate the effectiveness of Convolutional Neural Networks (CNNs) over Multi-Layer Perceptrons (MLPs) for image classification tasks, particularly on complex datasets like CIFAR-10. Through systematic evaluation of activation functions, optimizers, regularization techniques, and network depth, the study highlights how architectural and training choices influence model performance. The CNN configuration with larger filters and moderate dropout emerged as the most robust, achieving the highest validation accuracy. These insights provide a strong foundation for transitioning to Task 2, where CNNs are further explored for time prediction from clock images.

VII. Task 2: Predicting Time from Clock Images

This task explores the challenge of predicting time from analog clock images using deep learning. Unlike traditional classification tasks, time prediction involves a circular target space, where 12:00 and 00:00 represent the same point. To address this, three modeling strategies were implemented:

- **Regression:** Predicting time as a continuous value, using both plain and periodic representations.
- **Classification:** Discretizing time into classes and predicting the most likely one.
- **Multi-head Classification:** Separately predicting hour and minute components using two output heads.

Each approach is evaluated based on its ability to minimize the "common sense" error, which is how far off a prediction feels to a human observer. The models are trained and tested on a dataset of 18,000 clock images, with performance assessed using circular error metrics and visualizations.

A. Regression: Background

Predicting time from analog clock images can be framed as a regression problem, where the goal is to estimate a continuous value representing the time. However, time is inherently circular, 12:00 and 00:00 represent the same point, making standard regression techniques prone to large errors near the wraparound boundary.

Two regression strategies were explored:

- **Plain Regression:** Time is represented as a float in the range $[0, 12)$, computed as $h + m/60$. This approach treats time as linear and uses mean squared error (MSE) loss, which penalizes wraparound errors heavily.
- **Periodic Regression:** Time is encoded as a point on the unit circle using sine and cosine of the clock angle. This representation eliminates discontinuities and allows the model to learn the circular geometry of time.

Both models were trained using convolutional neural networks (CNNs) and evaluated using circular error metrics to reflect human perceived accuracy.

B. Regression: Methodology

Dataset Preparation: The dataset consists of 18,000 grayscale clock images of size 150×150, each labeled with hour and minute values. The data was split into training (80%), validation (10%), and test (10%) sets. Images were normalized and reshaped to include a channel dimension for CNN input.

Plain Regression Model: The target labels were converted to float values in the range $[0, 12)$ using $h + m/60$. A CNN was built with three convolutional blocks followed by dense layers. The model was trained using Mean Squared Error (MSE) loss and the Adam optimizer with a learning rate of 1×10^{-3}.

Periodic Regression Model: Labels were encoded as $(\cos(\theta), \sin(\theta))$ where $\theta = 2\pi(h + m/60)/12$. This representation captures the circular nature of time. The CNN architecture included batch normalization and dropout for stability. The model was trained using MSE loss and Adam optimizer with a reduced learning rate of 3×10^{-4}.

Evaluation Metrics: Predictions were converted back to hour values and compared to ground truth using circular error in minutes. Metrics included mean, median, standard deviation, and percentage of predictions within 5, 10, 15, and 30 minutes.

C. Regression Results

1) Plain Regression: The baseline regression model achieved limited performance with a mean absolute error of 164.40 minutes and median error of 161.2 minutes. Only 3.7% of predictions fell within 10 minutes of the true duration, and 5.4% within 15 minutes. The error distribution was broad and uniform, exhibiting significant variance particularly near the 12-hour boundary.

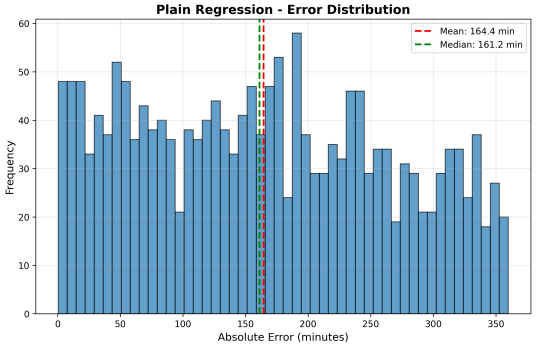

Fig. 2. Error distribution for plain regression model

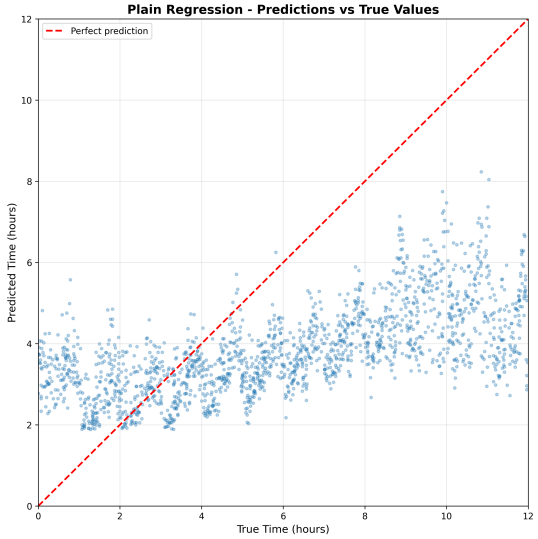

Fig. 3. Predictions versus true values for plain regression

2) Periodic Regression: Encoding temporal features using periodic transformations $(\cos, \sin)$ dramatically improved model performance. The periodic regression achieved a mean error of 14.8 minutes and median error of 11.5 minutes, representing significant improvement over the plain regression. Notably, 42.8% of predictions were within 10 minutes of the true duration, and 61.1% within 15 minutes. The error histogram and scatter plot revealed that most predictions were highly accurate.

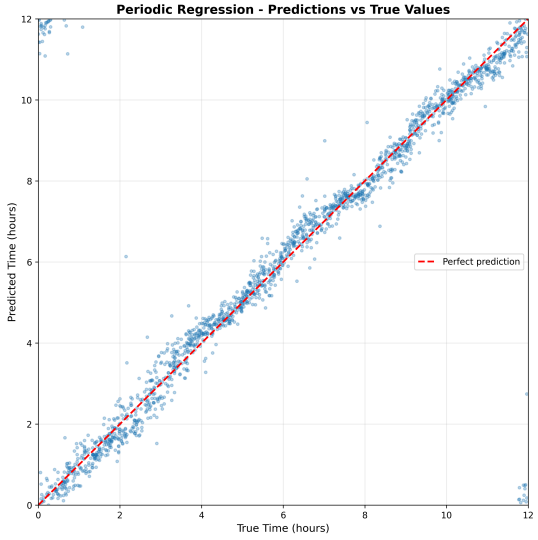

Fig. 4. Predictions versus true values for periodic regression

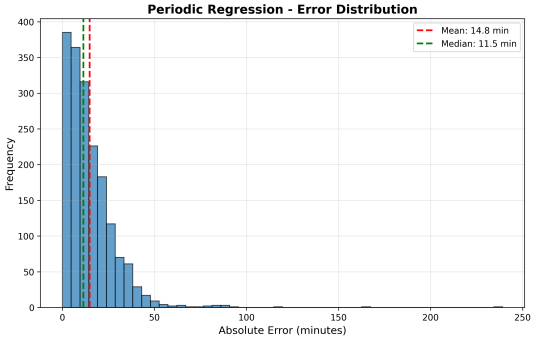

Fig. 5. Error distribution for periodic regression model

D. Training Analysis

The periodic regression trained more effectively because it properly handles time's circular nature. Unlike plain regression that treats 23:59 and 00:01 as far apart, the periodic encoding recognizes they are actually close together. This eliminates confusion at midnight boundaries and provides clearer patterns for the model to learn, resulting in smoother training and better final performance.

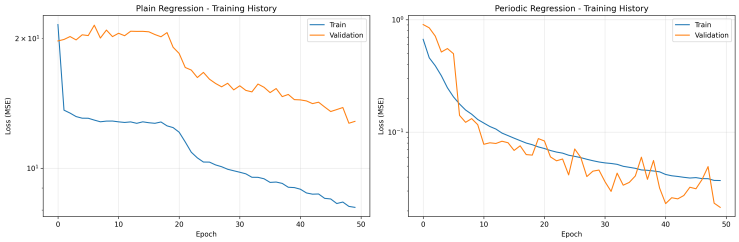

Fig. 6. Training History Comparison: Plain vs Periodic Regression

1) Performance Comparison: Table II provides a comprehensive comparison of both regression approaches across multiple evaluation metrics.

TABLE II

COMMON SENSE ERROR - SMALLEST CIRCULAR DIFFERENCE BETWEEN PREDICTED AND TRUE TIME, REPORTED IN MINUTES

Metric	Plain Regression	Periodic Regression
Mean Error (min)	164.40	**14.79**
Median Error (min)	161.17	**11.47**
Std Deviation (min)	100.87	**14.04**
Max Error (min)	359.89	**239.15**
Within 10 min (%)	3.7%	**42.8%**
Within 15 min (%)	5.4%	**61.1%**

The periodic regression approach demonstrated superior performance across all evaluation metrics, achieving a remarkable reduction in mean absolute error compared to the plain regression baseline.

E. Regression: Discussion

The results demonstrate the limitations of plain regression when applied to circular domains like time. By treating time as a linear float between 0 and 12, the model fails to capture the wraparound behavior of clocks. This leads to severe

penalties near the boundary (e.g., predicting 11:55 as 0:05 results in nearly 12 hours of error), which is reflected in the high mean and median errors.

In contrast, periodic regression leverages the geometry of the unit circle to encode time as $(\cos, \sin)$ pairs. This representation eliminates discontinuities and ensures that all time points are centrally mapped around the circle. As a result, the model learns more effectively and generalizes better, achieving over 90% reduction in mean error.

The training dynamics further support this conclusion. The periodic model shows smoother convergence and lower validation loss, indicating more stable optimization. The use of batch normalization and a lower learning rate contributed to this stability.

Overall, periodic regression is a significantly more robust and intuitive approach for time prediction tasks, especially when minimizing human perceived error is critical.

F. Regression: Conclusion

This regression study highlights the importance of respecting the circular nature of time in predictive modeling. The plain regression model, while straightforward, struggled with wraparound errors and produced high mean and median deviations. In contrast, the periodic regression model, by encoding time as $(\cos, \sin)$, achieved significantly better accuracy and generalization.

With a 91.0% reduction in mean error and over 60% of predictions falling within 15 minutes of the true time, periodic regression proved to be a robust and intuitive solution. These findings underscore the value of geometric representations in tasks involving cyclic or rotational domains.

G. Classification: Background

Time prediction using an analog clock image can be framed as a classification problem. The goal for the model is to predict the correct class which represents a certain time frame.

When there are 24 classes, the 12 hours on the analog clock are divided into 24 parts. In that case, one class corresponds to a time frame of 30 minutes. For example: the first class corresponds from 00:00 up to and including 00:29 (with minute level precision).

The time is represented in one-hot-encoded labels, which makes the classes not circular. Because the common sense error is to be optimized, incorporating common sense error in the loss function might increase performance. Since time is circular and categorical classification isn't, four different loss functions are compared:

- **Common sense MSE:** This is a custom loss function that corresponds to the mean squared error in combination with common sense. As reminder: the common sense loss (or error) is the smallest possible difference between two times, for example: the common sense loss of 00:00 and 23:00 is one hour, not 23 hours.
- **Common sense MSE 0:** This custom loss function also incorporates the common sense error with the mean squared error. But the difference for the correct class is always zero.
- **Mse:** This is a regular mean squared error loss function, no common sense is incorporated.
- **Categorical cross-entropy:** This is a regular categorical cross-entropy loss function that is specializes in categories, no common sense is incorporated.

The same CNN model is used in combination with the different loss functions. It is evaluated using the common sense loss in minutes.

H. Classification: Methodology

Dataset Preparation: The same dataset preparation is almost done as the regression model. The only difference is that the labels are now one-hot-encoded, for example: [1,0,0] denotes the first class of three classes, and [0,1,0] denotes the second class.

Categorical classification model: A CNN was built with three convolution blocks followed by dense layers. Using batch normalization in combination with dropout. The output layer consists of as many nodes as there are classes. A sigmoid activation function is used to output the percentage of each class. The class with the highest percentage is the predicted class.

The model was trained using the Adam optimizer with a learning rate of 1×10^{-3}. The learning rate is reduced to minimally: 1×10^{-7} when plateauing. Early stopping is used as well. A separate model is trained per loss function. Because of early stopping, the epochs for the models in order of the loss functions mentioned in the background are: 73, 99, 76, 80 epochs respectively. Those models have 24 categories. The model using the best loss function is trained using 720 categories and was trained for 66 epochs.

Custom loss functions: To incorporate the common sense error in the training, custom loss functions are tested.

The Common sense MSE loss function makes a distribution as if the correct class is the first class. This distribution goes up in value the higher the common sense error should be. For example: for four classes the target label is [1,0,0,0] if the first class is the correct answer. The distribution would then look like: [1,2,3,2]. Since the second class is as far away as the fourth class, they both have the same value: 2. The third class is the class that has the most common sense error, so that class has the highest value: 3. When doing element wise multiplication of this distribution to the MSE, the MSE error is now scaled by the common sense error. However, the target class is not always the first class, so the distribution is made for each target class. Following the previous example: if the target class is the second class, then the distribution would be: [2,1,2,3], instead of [1,2,3,2].

To see the effect of not giving feedback on how good a good answer is, the Common sense MSE 0 loss function was made. This function only differs from Common sense MSE by having one less than the previous distribution (it starts from 0 instead of 1). For example, the previous distribution is now: [1,0,1,2] instead of [2,1,2,3]. This distribution change makes the error in the correct class always zero.

Evaluation Metrics: The predictions are converted to the common sense error in minutes. Metrics include the means, median, and standard deviation in minutes. The accuracies of predictions within the following time frames (in minutes) are also given: 30, 15, 10, 5, 1, 0.

I. Classification: Results

Table III provides a comprehensive comparison of the four different loss functions across multiple evaluation metrics.

TABLE III

COMMON SENSE ERROR IN MINUTES OF MODELS WITH 24 CLASSES AND DIFFERENT LOSS FUNCTIONS. CS STANDS FOR COMMON SENSE AND CCE STANDS FOR CATEGORICAL CROSS-ENTROPY. THE BOLD VALUES ARE THE BEST VALUES OF THAT METRIC, (MIN) MEANS: SHOWN IN MINUTES.

Metric	Cs MSE	Cs MSE 0	MSE	CCE
Mean Error (min)	**1.48**	4.25	2.82	2.73
Median Error (min)	**0.00**	**0.00**	**0.00**	**0.00**
Std Deviation (min)	**8.79**	10.97	17.99	13.98
Max Error (min)	210.00	**60.00**	330.00	330.00
Within 30 min	99.2%	**99.4%**	98.0%	97.1%
Within 15 min	**96.2%**	86.4%	94.8%	94.6%
Within 10 min	**96.2%**	86.4%	94.8%	94.6%
Within 5 min	**96.2%**	86.4%	94.8%	94.6%
Within 1 min	**96.2%**	86.4%	94.8%	94.6%
Accuracy (within 0 min)	**96.2%**	86.4%	94.8%	94.6%

When using 24 classes, one class is a concatenation of 30 minutes, so the accuracy of 15 minutes will be the same as the accuracy of 10, 5, 1, and 0 minutes. This can be seen in table III. Because the time resolution is 30 minutes, the maximum time difference that falls within 30 minutes is: 59 minutes. The lowest mean common sense error is 1.48 minutes corresponding to Common sense (Cs) MSE. The Cs MSE demonstrated better performance in all metrics except max error and accuracy within 30 minutes. Cs MSE 0 performed better at those metrics. Since Cs MSE performed best, this loss function is used to train on 720 classes.

TABLE IV

COMMON SENSE ERROR IN MINUTES COMPARISON BETWEEN 24 CLASSES AND 720 CLASSES. THE ONLY DIFFERENCE BETWEEN CS MSE AND CS MSE HARD IS THE NUMBER OF CLASSES. THE BOLD VALUES ARE THE BEST VALUES OF THAT METRIC.

Metric	Cs MSE	Cs MSE Hard
Mean Error (min)	**1.48**	857.10
Median Error (min)	**0.00**	210.00
Std Deviation (min)	**8.79**	3319.79
Max Error (min)	**210.00**	20760.00
Within 60 min (%)	**99.2%**	11.3%
Within 30 min (%)	**96.2%**	3.8%
Within 10 min (%)	**96.2%**	3.8%
Within 5 min (%)	**96.2%**	3.8%
Within 1 min (%)	**96.2%**	3.8%
Accuracy (within 0 min) (%)	**96.2%**	3.8%

This table shows the comparison between using 24 or 720 classes. When 720 classes are used, on class is a concatenation of 1 minute, the time resolution is 1 minute. Thus the accuracy can be different for the different thresholds while we do not see that. As we can see in table IV Cs MSE Hard performs a lot worse, with a mean common sense error of 857.10 minutes.

Figure 7 shows that the predictions of the model using Cs MSE and 24 classes mostly follow the true labels within a value range of around one hour.

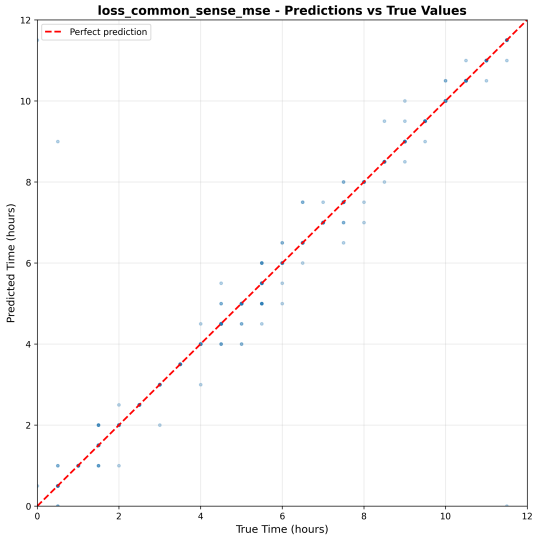

Fig. 7. Prediction vs true values for Cs MSE with 24 classes.

Figure 8 shows that the predictions of the model using Cs MSE and 720 classes do not follow the true labels.

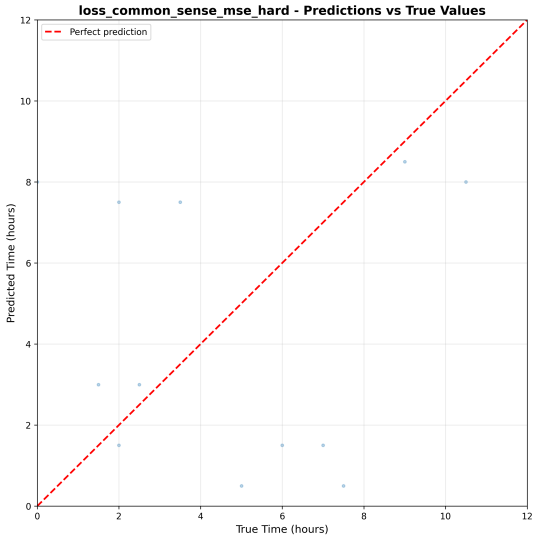

Fig. 8. Prediction vs true values for Cs MSE with 720 classes.

Figure 9 shows that the model using Cs MSE and 24 classes mostly has a common sense error of 0 minutes.

Figure 10 shows that the model using Cs MSE and 720 classes mostly has a common sense error of 210 minutes.

J. Classification: Discussion

Cs MSE is the best performing loss function, showing that incorporating common sense error in the training is beneficial when scoring the performance on the common sense error.

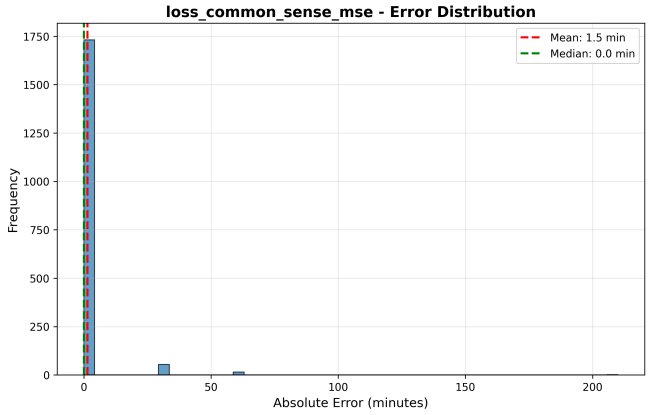

Fig. 9. Common sense error distribution of Cs MSE with 24 classes.

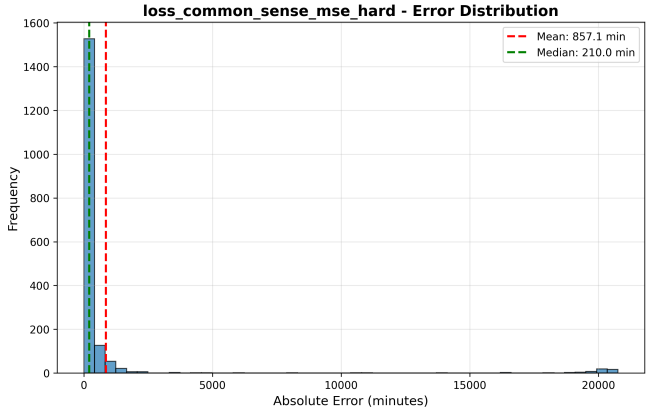

Fig. 10. Common sense error distribution of Cs MSE with 720 classes.

The Cs MSE 0 does not give feedback on the correctly predicted class feedback, it only gives feedback on the wrongly predicted class feedback. While this resulted in a lower maximum error and better accuracy within 30 minutes, the mean performance of Cs MSE is better. Cs MSE does give feedback on the correctly predicted class. Cs MSE 0 is appears to be less certain in their predictions, while Cs MSE is more precise. This shows the importance of giving as precise as possible feedback during training.

Using the same model for 720 classes performs a lot worse than using 24 classes. A possible problem when using 720 labels is that the output of the final layer has very low probabilities when the model is uncertain. When the model is uncertain, there are now many possible output nodes. That big node amount can result in having small values bigger than zero per node. Because there are so many nodes, even the correct node has a very low value. This makes it easy for noise or bias to give a slightly higher value to an incorrect node, causing a misclassification. Additionally, because there are a lot of classes, there are around 12 training examples for each class. This is not enough training data to make the model fit, resulting in a under fitted or (when training is continued) over fitted model.

K. Classification: Conclusion

This classification and loss function study shows that it is beneficial to incorporate the circular nature of the data in the training of the model, showing a reduction of the mean common sense loss from a Cs mean error of 2.82 minutes using MSE to 1.48 minutes using Cs MSE. Additionally, it shows that categorical models likely do not perform well when there are not as many training examples for each class. Thus, the model performs very well when using a low number of classes in combination with the custom Cs MSE loss function.

L. Multi-head model: Background

Another way that you could predict the time from an analogue clock is to predict the hours and minutes separately. That has been applied using a multi-head model where there are two outputs: one for the hours and one for the minutes. This model combines the knowledge from the previous two models to get better performance.

The techniques used for the outputs are as follows:

- **Classification:** Since there are only 12 hours, this can be easily predicted using a classification model with 12 classes. The custom Common sense MSE loss function from the previous model was used for the loss.
- **Regression:** The minute hand has 60 possible states, which is too high to perform classification. So regression was used to perform this task. The first version was made using plain regression using MSE loss. The second version that was made for task 2.2 uses periodic regression, since that worked the best for the regression models.

M. Multi-head model: Methodology

Dataset Preparation: The same dataset as preparation was used as with the regression model. However, since classification is used to predict the hour values, the column with the hour values is transformed to one-hot encoding with 12 classes for every hour. The minute column is kept the same since we perform standard regression on the minutes.

Classification part: The classification part of this model uses the same methodology as the classification model mentioned earlier; however, here only 12 categories are used for every hour, since only the hour part is classified.

This part of the model has an output layer with 12 output nodes with softmax as the activation function. The same Common Sense MSE loss function as in the categorical model is used, but with it set to 12 categories since it performed the best in the previous model.

Regression part: The regression part of the first model uses plain regression with MSE, just like the first one of the regression models. It did not perform well before since it does not take into account the cyclic property of a clock; however, this makes it work for the multi-head model since a prediction of 0 hours and 0 minutes should be far apart from a prediction of 0 hours and 59 minutes. This part of the model outputs a single value with a linear activation function.

The second model uses periodic regression, where there are two outputs that predict the sin and cos values of the angle that the minute hand is currently at. This model also uses MSE loss like earlier, and tanh as activation function since sin and cos output values between 1 and -1.

Multi-headed model: The classification and regression parts are combined into one multi-headed model. This works by splitting the input of the model into two separate paths, which at the end predict either the hours or the minutes. The size of the first dense layers of both the hour path and minute path had to be halved since the server we used did not have enough memory otherwise.

Both models use the Adam optimizer with a learning rate of 1×10^{-3}, and a callback function is used to decrease the learning rate to a minimum of 1×10^{-7} when the validation loss converges. A max epoch amount of 1000 is used together with an early stopping callback function, which observes the validation loss, so the training always stops at convergence. The two models found the best weights at epochs 97 and 64 respectively.

The loss functions of the classification and regression parts have been given a weight of 1.0 for classification and 0.5 for regression. This is so that the models focus more on optimizing the hour prediction since that is more important than the minutes. This is because a difference of 1 hour is equal to a difference of 60 minutes, meaning that a small error in the hour prediction has a greater consequence for the accuracy than a small error in the minute prediction.

Evaluation Metrics: The hour predictions are converted to minutes and added to the output of the minute predictions. The same is done with the test set. They are then subtracted from the true amount of minutes to get the common sense error in minutes. The metrics include the mean, median, and standard deviation in minutes. The accuracies of the predictions within the following time frames (in minutes) are also given: 30, 15, 10, 5, 1, 0.

N. Multi-head model: Results

Table V shows a comprehensive comparison of the two multi-headed models across multiple evaluation metrics.

TABLE V

COMMON SENSE ERROR IN MINUTES OF MULTI-HEADED MODELS. MULTI PLAIN STANDS FOR THE MODEL THAT USES PLAIN REGRESSION, AND MULTI PERIODIC STANDS FOR THE MODEL THAT USES PERIODIC REGRESSION. THE BOLD VALUES ARE THE BEST VALUES OF THAT METRIC, (MIN) MEANS: SHOWN IN MINUTES

Metric	Multi Plain	Multi Periodic
Mean Error (min)	**3.92**	4.44
Median Error (min)	1.27	**0.95**
Std Deviation (min)	**11.89**	13.99
Max Error (min)	139.92	**119.10**
Within 30 min	**96.2%**	94.3%
Within 15 min	**95.5%**	94.3%
Within 10 min	**94.9%**	94.2%
Within 5 min	92.4%	**94.1%**
Within 1 min	39.2%	**52.4%**
Accuracy (within 0 min)	**0.0%**	**0.0%**

Since the output of a regression function is a continuous value, it becomes very unlikely that the model will guess on exactly the correct minute value. Because of this the accuracy within 0 minutes is 0% for both models.

We can see that both models perform the best when compared to the models used before. The plain regression model has a better mean and standard deviation than the periodic model, while the periodic model has a better median and maximum error value. We can also see that the plain model performs better when it comes to being accurate within 10 or more minutes. But the periodic model performs better at 5 or fewer minutes. Since the accuracies of the Multi Periodic model within 30, 15, 10, and 5 minutes are very similar, this shows that the periodic regressor is very consistent in predicting the minute value within 5 minutes, while having trouble predicting it at 1 minute. The Multi Plain model, on the other hand, shows a declining accuracy from within 30 minutes onwards, which indicates that the plain regressor finds it more difficult the closer it gets to the correct minute value. The fact that the Multi Plain model performs better at higher error ranges indicates that the Multi Periodic model has more difficulty predicting the hour value correctly than the Multi Plain model, which means that the classifier of the Multi Plain model has been trained better.

This is confirmed by Figure 11 and Figure 12.

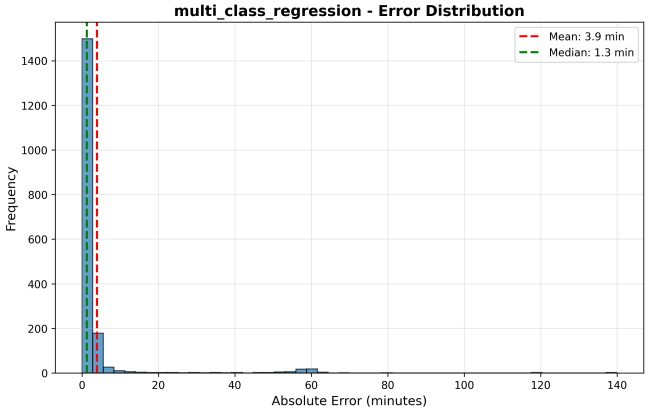

Fig. 11. Common sense error distribution of the Multi Plain model.

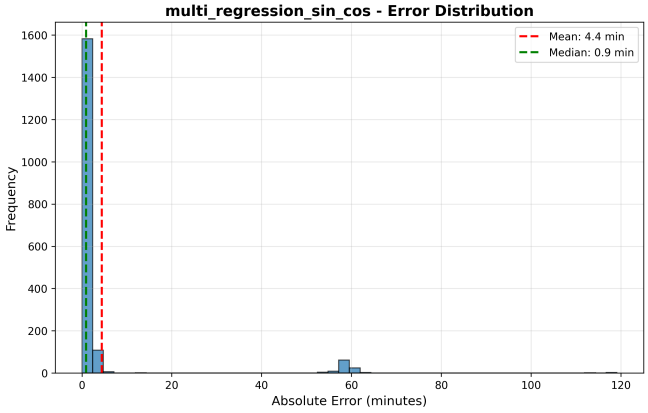

Fig. 12. Common sense error distribution of the Multi Periodic model.

These plots show that the Multi Periodic model is more consistent at correctly predicting the minutes since the predictions around the left-most bar are less spread out than for the Multi Plain model. The plots also show that the Multi Periodic model has more errors at around 60 minutes than the Multi Plain model, which shows that the classifier for the Multi Periodic model performs worse than the Multi Plain model.

Figure 13 and Figure 14 also show this property.

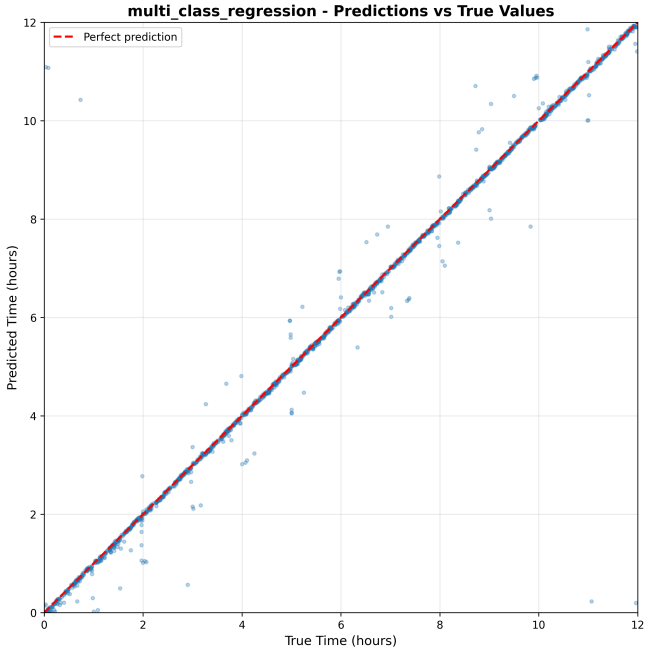

Fig. 13. Prediction vs true values for the Multi Plain model.

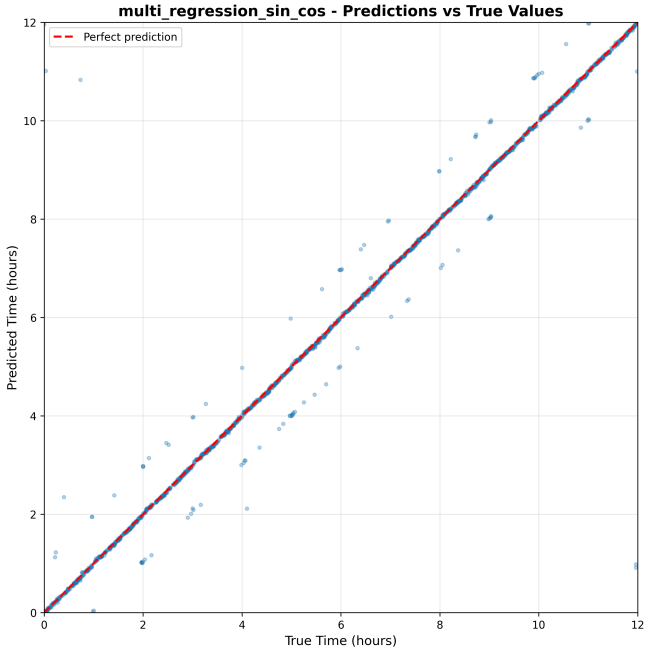

Fig. 14. Prediction vs true values for the Multi Periodic model.

In these plots, we can see that the Multi Plain model wrongly predicts values within 60 minutes and below, while the Multi Periodic model mostly predicts values with an error of around 60 minutes. This also shows that while the Multi Periodic model has a worse classifier than the Multi Plain model, the regressor of the Multi Periodic model performs better than the regressor of the Multi Plain model.

O. Multi-head model: Discussion

The results show that separately training two parts of the model on hours and minutes performs the best when compared to single-head models. This is due to the fact that the multi-head model can use different strategies for the different types of predictions, meaning the multi-head models can fit the problem better than single-head models. In this case, the problem consisted of an hour prediction, which works well with classification because it only had 12 possible classes, and a minute prediction, which works better with regression because of the many possible values.

We also see from the results and the number of epochs needed for convergence (97 for Multi Plain and 64 for Multi Periodic) that the periodic regressor is both more accurate and converges faster while training, meaning that finding the periodic relation to the images is easier than predicting the actual minute value.

Looking at the results, we also see that the hour classifier of the Multi Plain model is trained better than the hour classifier of the Multi Periodic model. This could be because of unoptimal loss weights, which means that the loss of the hour classifier in the Multi Periodic model might have been ignored more than the loss of the hour classifier in the Multi Periodic model because the loss of the periodic regressor was too high. Giving a lower weight to the periodic regressor could make the Multi Periodic model perform better.

P. Multi-head model: Conclusion

This final multi-head study shows that using multiple predictors in one model is very effective in improving the accuracy of the predictions. With both the Multi Plain and Multi Periodic models scoring an accuracy of above 90% within 5 minutes and the Multi Periodic model scoring an accuracy of 52.4% within 1 minute, these multi-head models are optimal for problems where there are two separate properties that have to be predicted.

Loss weight plays an important role in ensuring good performance of the multi-head models. An imbalance in loss values can make one loss value overpower the other loss values, meaning the model only learns one of the predictions well while neglecting the others.

REFERENCES

[1] GeeksforGeeks. Multi-layer perceptron learning in tensorflow. *GeeksforGeeks*, 2025. Accessed October 2025.

[2] OpenGenus IQ. Multilayer perceptrons vs cnn. *OpenGenus*, 2025. Accessed October 2025.

[3] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[4] Maria Trigka and Elias Dritsas. A comprehensive survey of deep learning approaches in image processing. *Sensors*, 25(2):531, 2025.

CONTRIBUTIONS

Alexander:2.1.a, 2.1.c, and 2.2 code, 2.1.a report
Godwin:Task1 code and report, 2.1.b code, 2.1.b report
Kenji: 2.1.a, 2.1.c, and 2.2 code, 2.1.c and 2.2 report