Vision and Recognition Project

Sara Francavilla

University of Trento, Trento TN, Italy
sara.francavilla@studenti.unitn.it
ID: 235951

Abstract. This report compares SVM on the Walmart temporal dataset with MLP and CNN on the Animals-10 image dataset, showing how CNNs effectively exploit spatial features to surpass MLPs, while SVM handles structured temporal data efficiently.

1 Classification on the Walmart Dataset

This section presents a classification task on a structured dataset with features reflecting retail or temporal dynamics. The objective is to categorize each instance by temporal labels—either by month or by season. These labels differ in granularity, and the goal is to assess how well the dataset supports both coarse (seasonal) and fine (monthly) classification.

A Support Vector Machine (SVM) is used as the classifier to examine how the data structure enables distinction across these temporal levels.

A Support Vector Machine (SVM) is a supervised learning algorithm used for classification. It determines the optimal hyperplane that maximizes the margin between classes, defined by the closest data points—support vectors, enhancing generalization and robustness.[1]

1.1 Dataset Structure

The dataset contains information from various stores, including the date of data collection, weekly sales, holiday flags, temperature, fuel price, CPI, and unemployment. To prepare these features for classification, numerical parameters are normalized to ensure a uniform scale and support model convergence. For temporal categories such as month and season, cardinal numbers are used instead of categorical labels in order to help the model capture temporal continuity. Although this approach overlooks the cyclical nature of time, it may enhances performance compared to standard labeling given an informative structure.

Plotting monthly variations reveals temperature's strong seasonal pattern, correlating well with months and seasons. Fuel price shows month-related trends, adding some predictive value, while CPI and unemployment vary little over time,

[1] For a detailed theoretical foundation, see Cortes & Vapnik's seminal work [1].

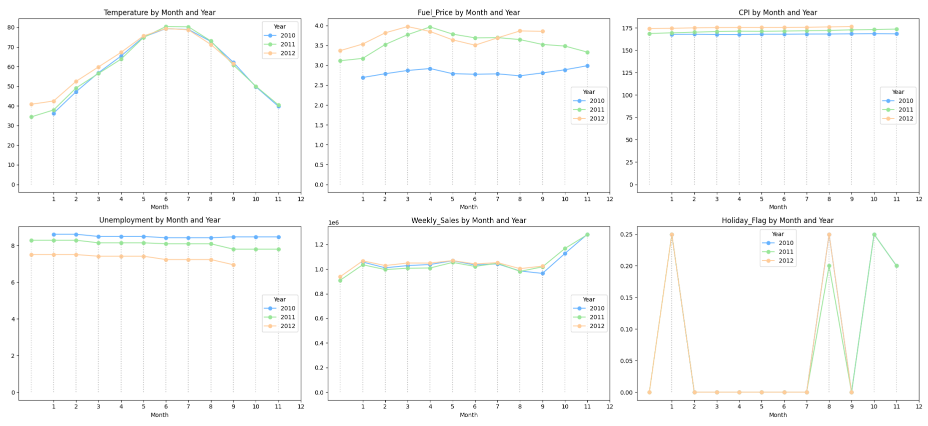


Fig. 1. Feature-months relations among the years

contributing less to classification. Weekly sales spike near holidays, especially year-end, offering useful temporal signals.

Overall, no single feature dominates, reducing overfitting risk and enhancing generalizability. Since stores operate independently across the full year, splitting data by store effectively prevents temporal or contextual leakage in training, validation, and testing.

1.2 The normalization process

Normalization is essential for classifiers like SVMs sensitive to feature scale. Continuous features are normalized (e.g., zero mean, unit variance) to ensure balanced influence and better convergence. Categorical features, being non-numeric, use one-hot encoding to prevent false ordinal assumptions. [2].

One-hot encoding is applied only to the year feature, as its numeric values lack meaningful continuity for classification. The store feature is excluded since the dataset is split by store, rendering it constant and irrelevant within each subset. The holiday flag is binary and requires no transformation.

1.3 The SVMs

SVM types differ by kernel choice, each suited to certain data patterns. The RBF kernel, the one chose for this dataset, effectively models complex non-linear boundaries by mapping data into higher-dimensional space, offering greater flexibility than linear or polynomial kernels.

Key hyperparameters—*C*, *gamma*, and *class_ weight*— control the model behavior. *C* balances training accuracy against model complexity, with higher values reducing regularization. *Gamma* sets the influence range of support vectors, where lower values produce smoother decision boundaries. *class_weight='balanced'* addresses class imbalance by weighting classes inversely to their frequency [3].

C	gamma	class_weight	month_acc	season_acc
1	'scale'	NONE	0.41	0.66
10	0.1	NONE	0.46	0.65
100	0.1	NONE	0.43	0.68
1000	0.01	NONE	0.49	0.68
1000	0.01	'balanced'	0.49	0.69

Table 1. Accuracy based on parameters and classification

Higher *C* combined with lower *gamma* yielded better accuracy in the complex month classification, reflecting smoother yet sufficiently flexible boundaries. The modest improvements from balanced class weighting indicate mild class imbalance. These outcomes align well with the theory of SVM kernel behavior and hyperparameter effects.

1.4 Result comparison

The differing accuracies between month (49%) and season (69%), as shown in table 1, highlight the varying complexity of the tasks and the role of SVM hyperparameters in managing this complexity. Season classification benefits from clearer, more separable patterns, while month classification involves subtler distinctions and overlapping classes.

Increasing C reduced underfitting by allowing the model to fit complex patterns, especially for the month task, while lowering gamma smoothed decision boundaries by expanding support vector influence, preventing overfitting. Excessively high C or gamma caused overfitting, harming validation, whereas too low values led to underfitting and poor performance.

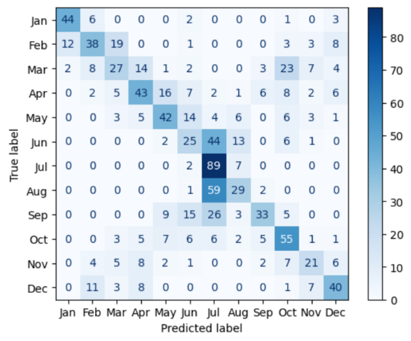


Fig. 2. Month prediction

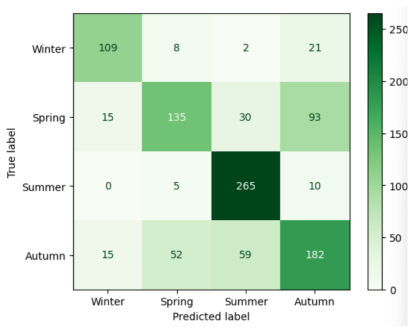


Fig. 3. Season prediction

The results show some confusion between adjacent months and similar seasons, likely due to gradual feature changes like temperature and CPI. However, the holiday flag improves distinction in winter, and extreme summer temperatures provide strong signals, making summer the easiest season to identify.

Proper feature scaling and encoding, along with careful hyperparameter tuning, mitigate overfitting and improve generalization, especially in complex classification problems like month prediction.

2 Classification on the Animal Dataset MLP

This work tackles a multiclass image classification task using the Animals-10 dataset, comprising images labeled across 10 animal classes. The goal is to build a model that accurately assigns each image to its corresponding category based on visual features. To handle varying image sizes, all inputs are resized to a fixed resolution and pixel values are normalized. A simple Multilayer Perceptron (MLP) is used as neural network for the classification.

Neural networks consist of layers of neurons that apply linear transformations followed by non-linear activation functions. Trained on labeled data, they use backpropagation and gradient descent to minimize prediction error.

The, here used, Multilayer Perceptron (MLP) is a feedforward neural network with fully connected input, hidden, and output layers. It models complex patterns through stacked non-linear transformations. While simpler than convolutional networks, it is effective when spatial structure is less relevant, especially with flattened image inputs.[5]

2.1 Parameters and Hyperparameters

The choice of hyperparameters is guided by dataset characteristics and Colab's hardware constraints. The training is performed on Google Colab, with an NVIDIA Tesla T4 GPU, which limits available VRAM. Therefore, the batch size is set to 32 to balance training stability and memory usage.

The learning rate is fixed at 0.0001 to ensure smooth convergence, as MLPs can be sensitive to high learning rates due to their fully connected structure.

All images are resized to 128×128 with 3 RGB channels to standardize input and reduce computational cost. The mean and standard deviation used for normalization are computed per channel across the dataset. This normalization centers and scales pixel values, improving training speed and stability.

The dataset is split into 70% training, 15% validation, and 15% testing to ensure fair evaluation. The hidden dimension is set to 64 neurons, offering enough capacity for pattern learning while minimizing overfitting.

All values reflect a trade-off between performance, computational efficiency, and model generalization.

2.2 The Datamodule

The DataModule cleanly separates data handling from model logic, improving readability, scalability, and reproducibility. The *prepare_ data()* method initializes the dataset, while setup() splits it into training, validation, and test sets. The *train_ dataloader(), val_ dataloader()*, and *test_ dataloader()* methods return DataLoaders with consistent batch size. This structure reduces code repetition, supports seamless integration with PyTorch Lightning's training loop, and allows for easy modification of datasets or parameters without changing the model code [6].

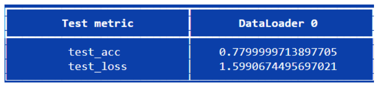
2.3 The Neural Network

Neural networks vary in architecture based on the data and task. The chosen model is a multilayer perceptron (MLP), a feedforward network with fully connected layers, suitable for fixed-size, flattened input like resized images. This MLP flattens the 128×128 RGB images—resulting in 49,152 input features—and passes them through two hidden layers with ReLU activations, in which the representation is more and more condensed(49,152 - 980 - 123), and dropout, followed by a LogSoftmax output layer for classification.

This structure processes all pixels simultaneously, learning non-linear mappings from images to labels without using spatial information. It balances simplicity and effectiveness, offering sufficient capacity for meaningful representations while being computationally efficient—well-suited to the dataset size, classes, and Colab GPU resources. More complex CNNs could improve accuracy by leveraging spatial features but demand much higher computational cost and training time.

2.4 Results Analysis

The MLP test results reveal an accuracy of 78% and a loss of 1.6, indicating that the model correctly classifies the majority of test samples but with moderate prediction confidence. These metrics reflect adequate performance given the task complexity and dataset characteristics, establishing a reasonable baseline for the MLP's effectiveness.



```
      Test metric            DataLoader 0
        test_acc          0.7799999713897705
       test_loss          1.5990674495697021
```

Fig. 4. Accuracy and losses

For the MLP, both ground truth and prediction samples include 10 elements from the test set. The number of correct predictions per class varies between 7 and 16, while incorrect predictions range from 0 to 6.

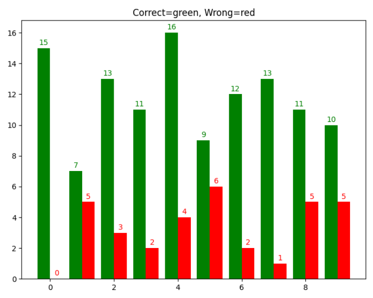


Fig. 5. Final result plot

This pattern demonstrates that the model accurately classifies several classes with minimal errors, though some classes exhibit higher misclassification. Overall, the model maintains a reasonable balance between correct and incorrect predictions, indicating consistent performance across most classes..

3 Classification on the Animal Dataset CNN

The key difference between an MLP and a CNN is how they process input data: MLPs flatten inputs into vectors, ignoring spatial structure, and use fully connected layers to learn patterns, while CNNs preserve spatial relationships using convolutional filters that capture local and hierarchical features. MLPs are simpler and computationally lighter, suitable for small datasets or naturally flattened inputs, but they often perform poorly on images due to ignoring spatial context, requiring more parameters and risking overfitting. CNNs exploit spatial structure to learn more meaningful features with fewer parameters, resulting in better accuracy and generalization for vision tasks, though they demand greater computational resources and longer training times [5].

This means that with respect to the MLP implementation we do expect to have better result starting from the same conditions.

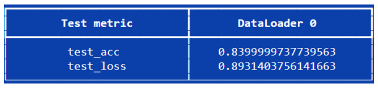
3.1 The CNN

The CNN implemented is a compact yet effective architecture designed for 128×128 RGB image classification. Its structure, with two convolutional layers followed by ReLU and max pooling, efficiently extracts and condenses spatial features while preserving important visual patterns. The large flattened feature space (65,536 units) allows the network to learn rich and detailed representations, which are then processed through fully connected layers with dropout to enhance generalization. The use of LogSoftmax ensures stable training for multi-class classification.

This design balances model complexity and computational efficiency, making it appropriate for datasets like Animals-10 and resource constraints such as the Colab GPU environment.

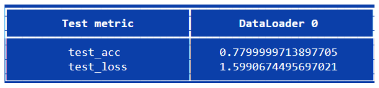
3.2 Results Comparison

The results highlight a clear performance difference between the CNN and MLP models on the test set. The CNN achieves a higher accuracy of approximately 84% with a lower test loss around 0.89, indicating more precise and confident predictions. In contrast, the MLP attains a lower accuracy of 78% and a higher loss of 1.6, reflecting less accurate classifications and lower confidence.



```
Test metric        DataLoader 0
test_acc           0.8399999737739563
test_loss          0.8931403756141663
```

Fig. 6. Accuracy CNN



```
Test metric        DataLoader 0
test_acc           0.7799999713897705
test_loss          1.5990674495697021
```

Fig. 7. Val-Train comparison MLP

While both models perform reasonably given the task complexity and dataset, the CNN demonstrates superior effectiveness in minimizing errors and correctly classifying samples. These metrics suggest that the CNN provides a stronger baseline, whereas the MLP may require further optimization or additional data to close the performance gap.

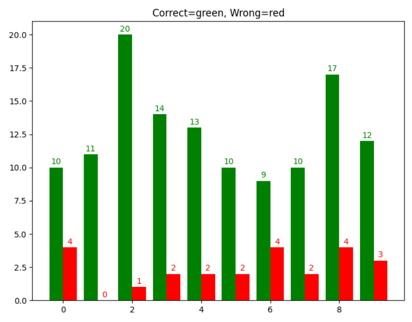


Fig. 8. Testing results CNN

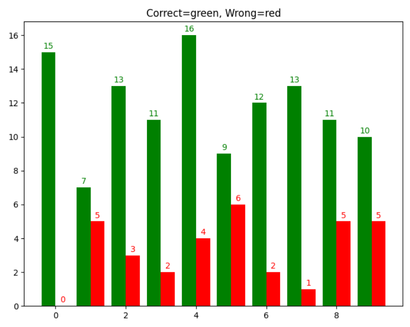


Fig. 9. Testing results MLP

The CNN demonstrates higher overall accuracy, with most classes showing strong precision and only a few misclassifications, indicating effective learning and reliable discrimination across categories. In contrast, the MLP exhibits more variability in performance across classes, with correct predictions ranging from 7 to 16 and misclassifications between 0 and 6 per class. While the MLP performs well on certain classes, it struggles with others, leading to a less consistent accuracy profile. Overall, the CNN delivers more robust and consistent classification results, whereas the MLP shows a reasonable balance but with greater room for improvement, especially on the classes where errors are more frequent.

References

1. Cortes, C., Vapnik, V.: Support-vector networks. *Machine Learning* **20**(3), 273–297 (1995). https://doi.org/10.1007/BF00994018
2. Jain, A., Nandakumar, K., & Ross, A.: Score normalization in multimodal biometric systems. Pattern Recognition **38**(12), 2270–2285 (2005)
3. Schölkopf, B., Smola, A., & Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation **10**(5), 1299–1319 (1998)
4. Hsu, C.W., Chang, C.C., & Lin, C.J.: A Practical Guide to Support Vector Classification. Technical report, Department of Computer Science, National Taiwan University (2010). Available at: https://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf
5. Goodfellow, I., Bengio, Y., & Courville, A.: Deep Learning. MIT Press (2016). Available at: https://www.deeplearningbook.org
6. Falcon, W. et al.: PyTorch Lightning. GitHub Repository. Available at: https://github.com/Lightning-AI/lightning (2023)