Geometric Shape Classification and Counting

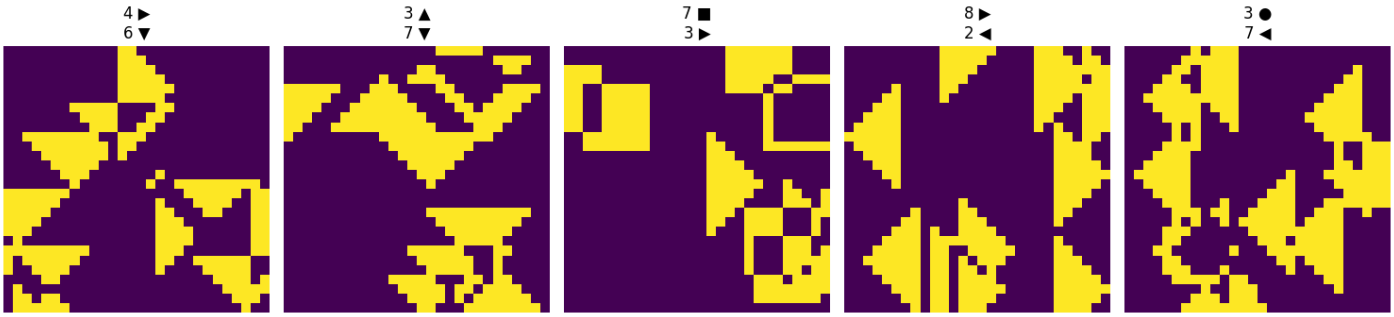
Wiktor Kotala

1 Introduction

This report documents the implementation and evaluation of a **multitask convolutional neural network** that performs classification and regression tasks simultaneously. The primary objective of this study is to analyze the effects of multitask learning on model performance and training dynamics.

2 Data and Augmentations

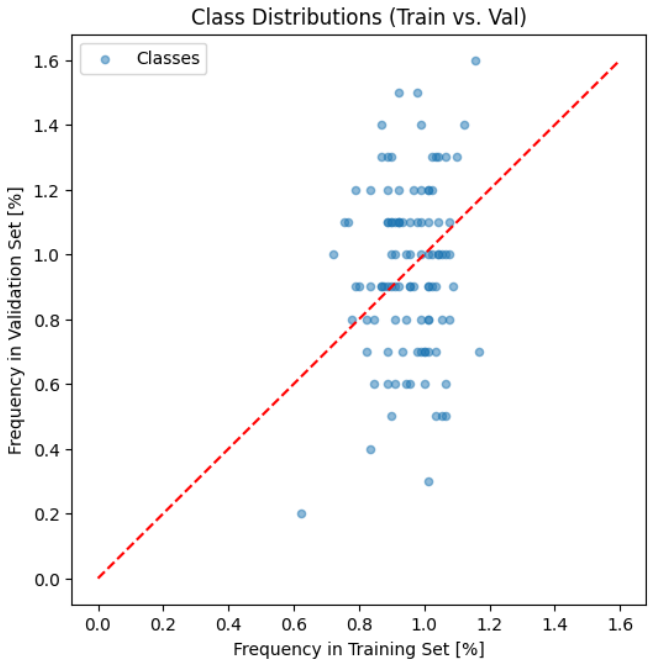
The Geometric Shape Numbers dataset consists of 10 000 images, split into 9 000 training samples, and 1 000 validation samples. Examples of images along with their class labels are shown below.



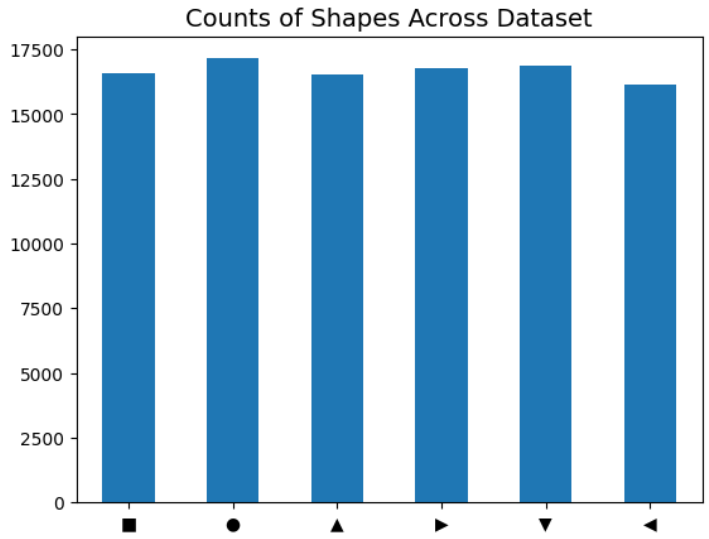
Sample images from the GSN dataset with ground truth labels.

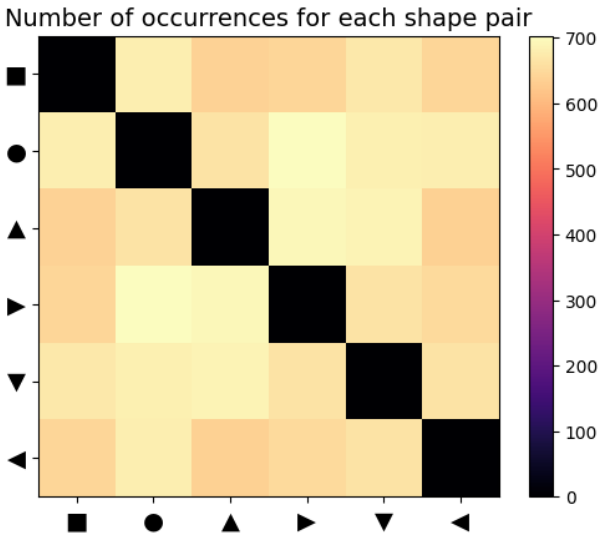
2.1 Exploratory Data Analysis

Analysis of the data reveals that only 105 out of 135 classes appear in the provided dataset. However, each of these 105 classes appear in both the training and validation sets. In the following graph each blue dot represents a class that appears in the GSN dataset. It clearly shows most of the classes appear with similar frequency in validation and training sets.



If we zoom out, the global statistics manifest the dataset is balanced.





2.2 Augmentation Strategy

To mitigate overfitting observed during initial testing of the network, and to partially address the issue of missing classes, I introduced 3 types of augmentations. Every epoch, each training sample passes independently through a pipeline consisting of:

1. horizontal flip with 50% probability,
2. vertical flip with 50% probability,
3. rotation by a random angle from set $\{0, 90, 180, 270\}$.

These augmentations were chosen because they change what types of shapes appear in the image, hence change the distribution of classes, possibly introducing new ones. This required careful implementation to change class labels and permute regression targets.

3 Methodology

3.1 Model Architecture

The model consists of the following backbone and two seperate heads for classification and regression tasks:

```
self.backbone = nn.Sequential( # Feature extractor
    nn.Conv2d(1, 8, 3, stride=1, padding=1), nn.ReLU(),
    nn.Conv2d(8, 16, 3, stride=1, padding=1), nn.ReLU(),
    nn.Conv2d(16, 32, 3, stride=1, padding=1), nn.ReLU(),
    nn.Conv2d(32, 64, 3, stride=1, padding=1), nn.ReLU(),
    nn.Flatten(start_dim=1),
    nn.Linear(64 * 28 * 28, 256), nn.ReLU()
)

self.head_cls = nn.Sequential( # Head for classification task
    nn.Linear(256, 256),
    nn.ReLU(),
    nn.Dropout(),
    nn.Linear(256, 135),
    nn.LogSoftmax(dim=1)
)

self.head_cnt = nn.Sequential( # Head for regression task
    nn.Linear(256, 128),
    nn.ReLU(),
    nn.Dropout(),
    nn.Linear(128, 6),
)
```

3.2 Loss Function

The network is optimized using a joint loss function that combines classification and regression objectives:

$$\mathcal{L} = \lambda_{cls} \cdot \mathcal{L}_{cls}(\hat{y}_{cls}, y_{cls}) + \lambda_{cnt} \cdot \mathcal{L}_{cnt}(\hat{y}_{cnt}, y_{cnt})$$

where $\hat{y}_{cls}$ and y_{cls} denote the predicted log-probabilities and ground truth class indices, while $\hat{y}_{cnt}$ and y_{cnt} represent the predicted and actual shape counts vectors. The hyperparameters λ_{cls} and λ_{cnt} balance the task importance. We utilize **Negative Log-Likelihood Loss** ($\mathcal{L}_{cls}$) for the classification task and **SmoothL1Loss** ($\mathcal{L}_{cnt}$) for regression.

4 Experiments

4.1 Setup

I optimized the model using **Adam** ($lr = 10^{-3}$) with a batch size of 64. Training ran for a maximum of 100 epochs, utilizing **early stopping** (patience = 10) monitoring the validation loss to prevent overfitting.
I evaluated the model under three distinct configurations to isolate the effects of multitask learning:

1. **Classification-Only**: $\lambda_{cls} = 1, \lambda_{cnt} = 0$. The model ignores regression targets.
2. **Regression-Only**: $\lambda_{cls} = 0, \lambda_{cnt} = 1$. The classification head is ignored.
3. **Multitask**: $\lambda_{cls} = 1, \lambda_{cnt} = 4$. The model optimizes both objectives simultaneously. The value of $\lambda_{cnt} = 4$ was chosen to balance the initial magnitude of the regression loss against the classification loss, ensuring both tasks contribute equally to the gradient updates.

4.2 Results

The following table summarizes the performance of the three experimental configurations:

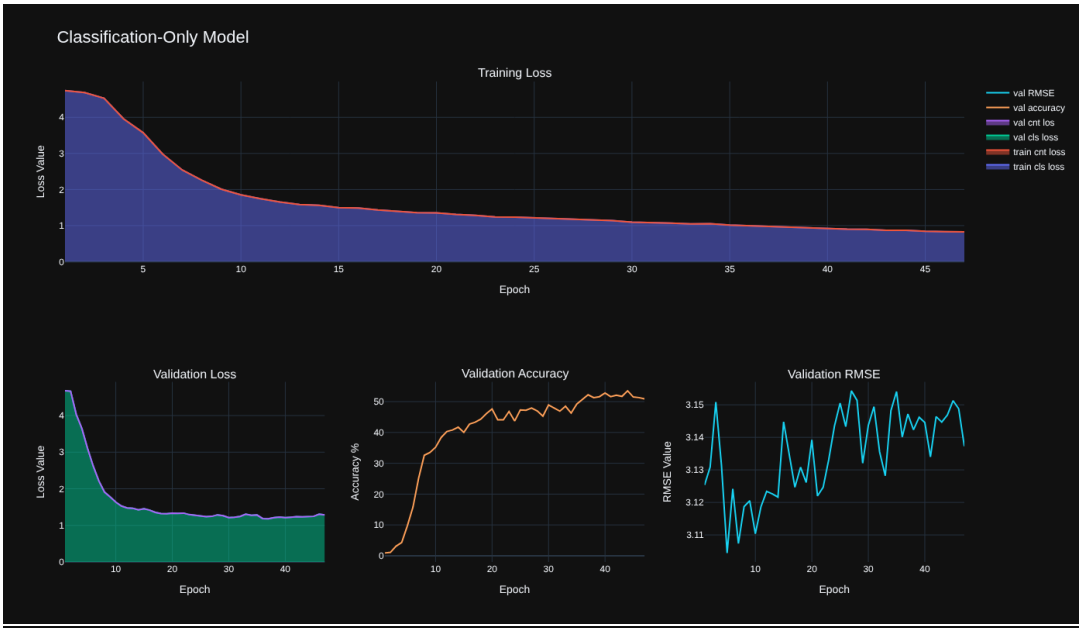
Metric	Classification-Only	Regression-Only	Multitask
Validation Loss	1.18	0.15	1.85
Accuracy	52%	-	50%
RMSE	-	0.59	0.61
Best Epoch	37	34	36
Runtime	7m56s	7m9s	7m26s

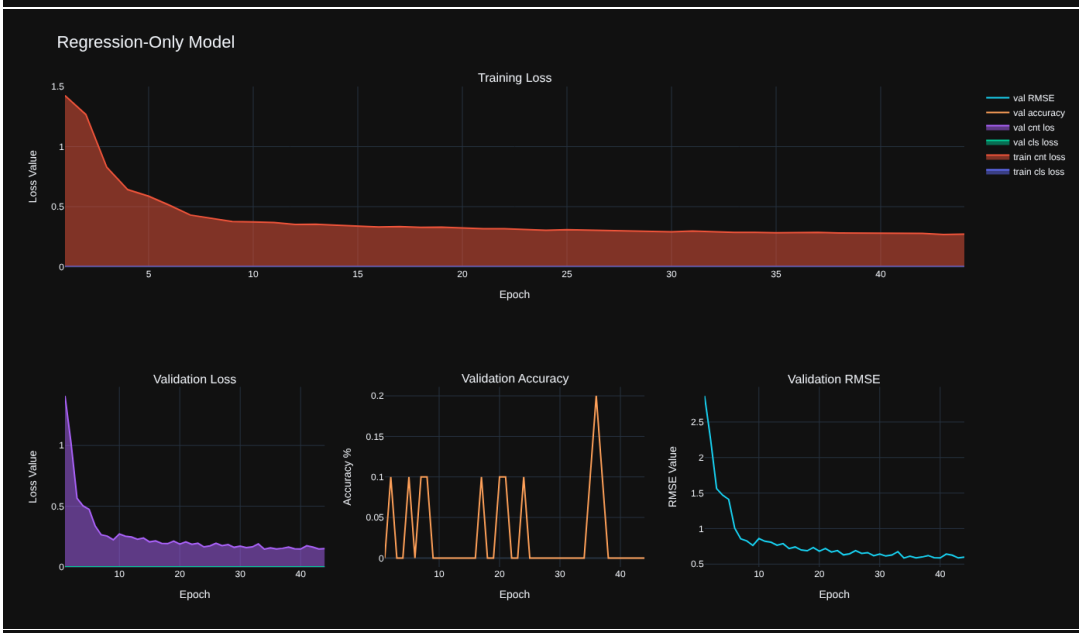
Notes: Values are reported from the epoch achieving the lowest validation loss. Losses are not normalized

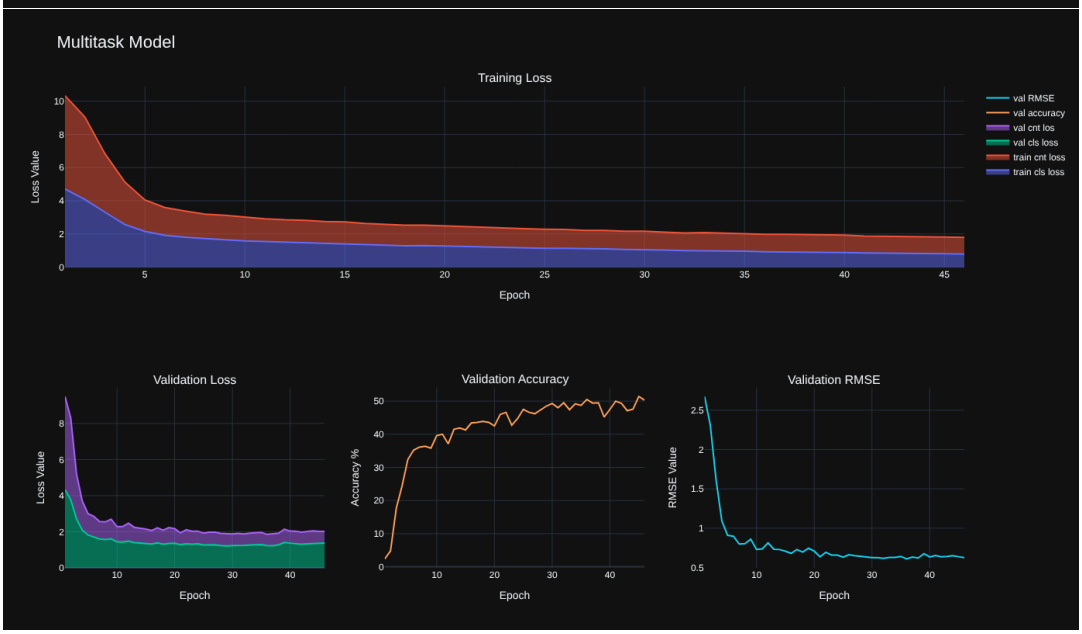
Graphs on the next (last) page display the learning curves for each experiment.
Although the code generates more detailed reports with additional metrics, they are omitted here as they are not essential for the final conclusions.

4.3 Conclusions

The experiments show that the Multitask architecture offers a trade-off between performance and efficiency. While the Multitask model showed a marginal performance drop compared to the single-task alternatives, it successfully learned both objectives within the same runtime and number of epochs. Hence, the most significant benefit of using Multitask approach is computational efficiency.
All models showed consistent training dynamics, and adding auxiliary tasks did not destabilize convergence nor did it require longer training.
Error analysis suggests that the fixed backbone capacity was the limiting factor. The confusion matrix and regression metrics suggest 'off-by-one' counting errors as the primary reason of failure.


Classification-Only Model
Training Loss
Loss Value
Epoch
val RMSE
val accuracy
val cnt los
val cls loss
train cnt loss
train cls loss
Validation Loss
Validation Accuracy
Accuracy %
Validation RMSE
RMSE Value


Regression-Only Model
Training Loss
Loss Value
Epoch
val RMSE
val accuracy
val cnt los
val cls loss
train cnt loss
train cls loss
Validation Loss
Validation Accuracy
Accuracy %
Validation RMSE
RMSE Value


Multitask Model
Training Loss
Loss Value
Epoch
val RMSE
val accuracy
val cnt los
val cls loss
train cnt loss
train cls loss
Validation Loss
Validation Accuracy
Accuracy %
Validation RMSE
RMSE Value