From Simulation to Reality with Random Noise

Steven Luo[1], Rory Lipkis[2], Ignacio López-Francos[3], Pavlo Vlastos[3], and Adrian Agogino[2]

[1] *University of California, Berkeley*

[2] *NASA Ames Research Center*

[3] *KBR / NASA Ames Research Center*

About

- Third-year at UC Berkeley
  - Computer Science, Data Science, Public Policy
  - Website: stevenfluo.github.io

Simulation-to-Reality

Real data can be

- ❌ Hard to gather
- ❌ Resource-intensive
- ❌ Expensive
- ❌ Time-consuming
- ❌ Potentially unsafe

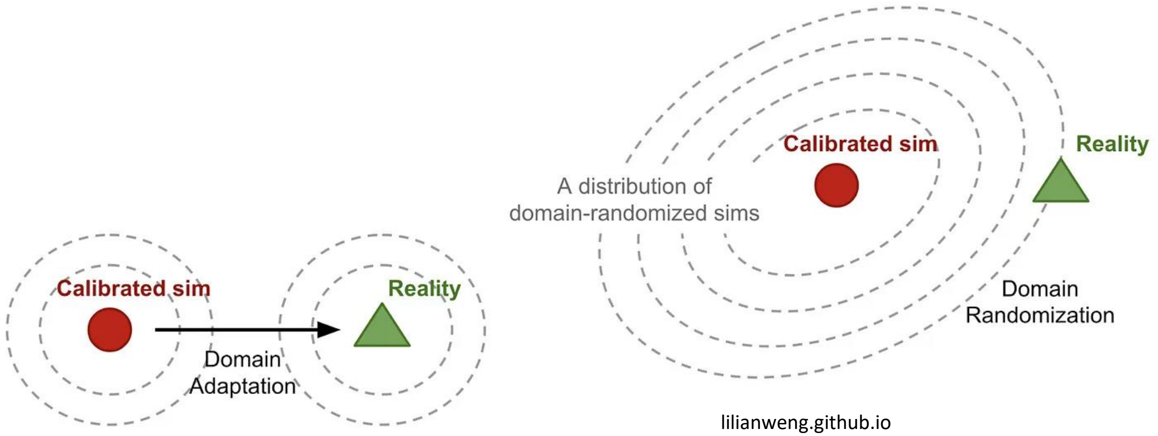
Simulated data is

- ✅ Easy to gather large volumes
- ✅ Low-cost
- ✅ Fast
- ✅ No threats to physical health

Zhao, W., Queralta, J. P., & Westerlund, T. (2020, December). Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In 2020 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 737-744). IEEE.

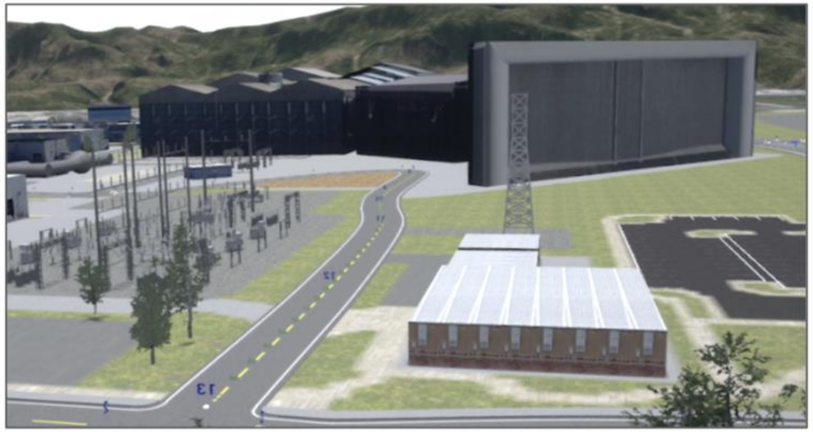
Simulated Data Isn’t Perfect

- “Reality gap” — physical and visual differences
- Methods for sim2real transfer
  - Domain/dynamics randomization
  - Simulation environments
  - Domain adaptation



lilianweng.github.io

AmesDT and RRAV



Lopez-Francos, I. G., Mitchell, S. C., Lipkis, R., Vlastos, P. G., Mbaye, S., & Infeld, S. I. (2023). A Model-Based Systems Engineering Approach for Developing an Autonomous Rover Testbed. In AIAA SCITECH 2023 Forum (p. 1894).

Research Goals

1. Use the RRAV testbed (rover) to conduct experiments
2. Validate Ames Digital Twin (AmesDT) as a good simulation of the Ames Research Center
3. **Quantify the effect of domain randomization on the real-world accuracy of classifiers trained in simulation**

Data

Figure 1: Images from AmesSim (left) and the real_data dataset (right).

Figure 2: Sample images from each dataset. Clockwise from top left: none_sim, sharpcj_sim, shot_sim, and the sharp_sim dataset.

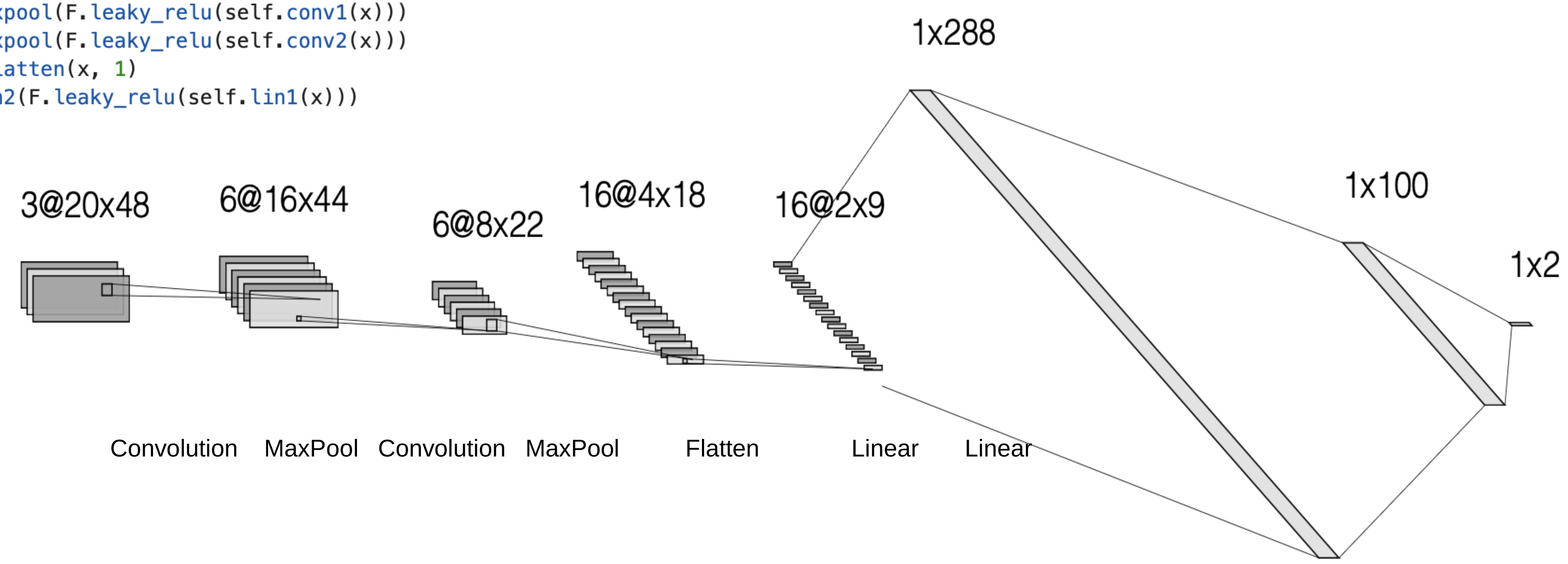
Real Data

Model Information: Left/Right Classifier

```python
class ConvNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)   # 5x5, 6 channels
        self.maxpool = nn.MaxPool2d(2, 2) # 2x2 max pool
        self.conv2 = nn.Conv2d(6, 16, 5)  # 5x5, 16 channels
        self.lin1 = nn.Linear(288, 100)   # fully connected to hidden layer
        self.lin2 = nn.Linear(100, 2)     # fully connected to logit output

    def forward(self, x):
        x = self.maxpool(F.leaky_relu(self.conv1(x)))
        x = self.maxpool(F.leaky_relu(self.conv2(x)))
        x = torch.flatten(x, 1)
        x = self.lin2(F.leaky_relu(self.lin1(x)))
        return x
```

Hyperparameters
Batch size: 128
Learning rate: 1e-3
Epochs: 25
Adam optimizer



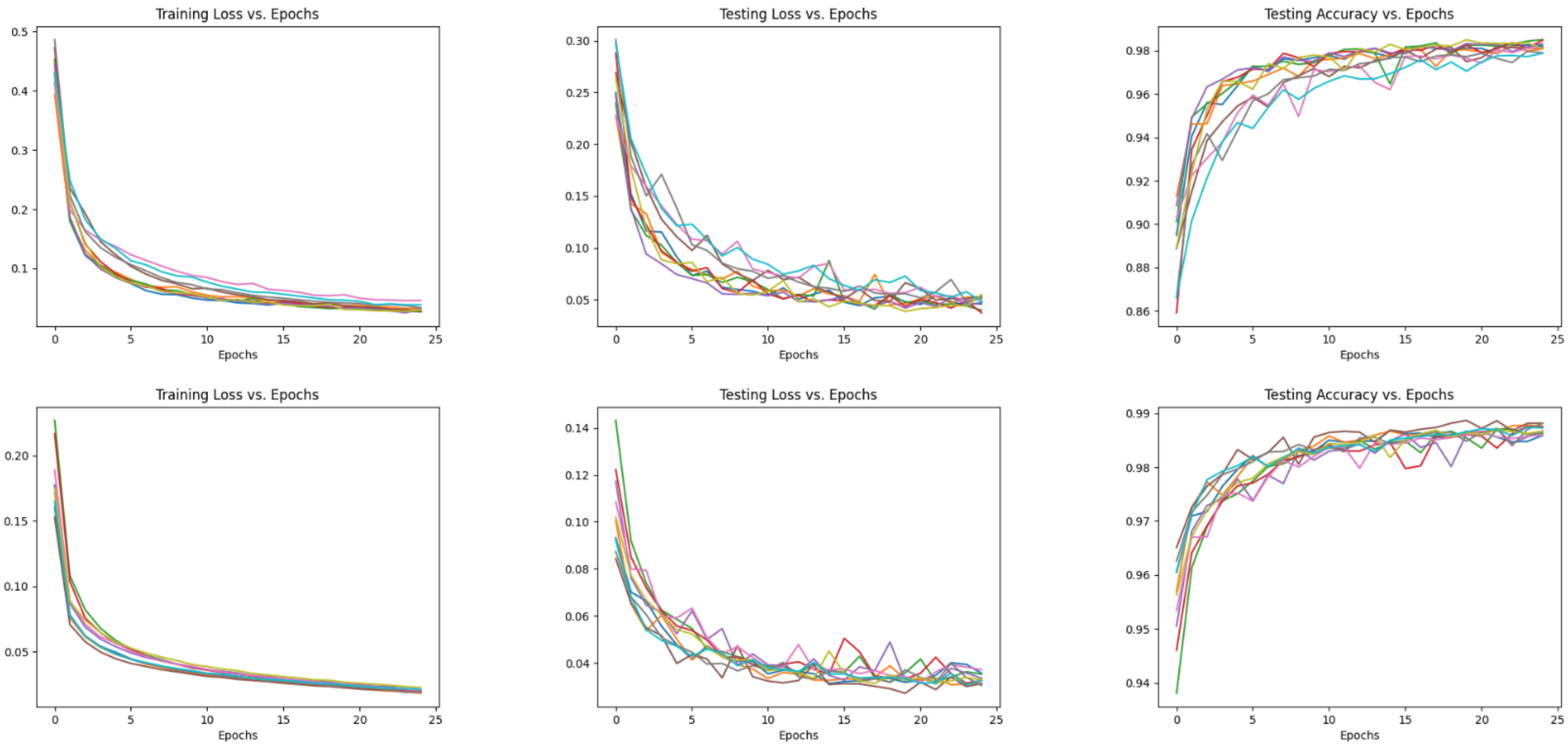
Results

Dataset	Size	Details	Data2Data	Data2Sim	**Sim2Real**
`none_sim`	48,000	No transforms	98.03%	N/A	**64.10%**
`sharp_sim`	528,000	Blur	99.75%	99.78%	**69.55%**
`sharpcj_sim`	528,000	Blur + Color Jitter	99.92%	99.94%	**71.80%**
`shot_sim`	528,000	Blur + Color Jitter + Poisson Noise	98.59%	99.51%	**<u>78.59%</u>**

Real dataset: 4,800 images captured in parking lot outside N269 using the RRAV testbed's camera

Real2Real average testing accuracy: <u>88.90%</u>

Results



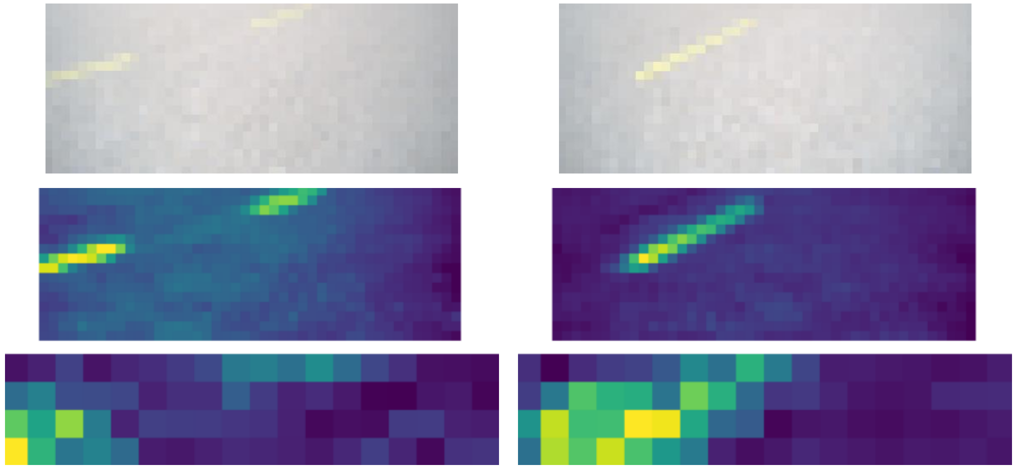
Performance metrics for models trained on the unaugmented none_sim dataset (top row) and the fully-augmented shot_sim dataset (bottom row).

Discussion

- Well-selected augmentations can bridge the reality gap, improve performance and classifier robustness
- Validated AmesDT and RRAV testbench for sim-to-real transfer learning using domain adaptation
- Applications include use as low-fidelity runtime monitor for airplane taxiing/centerline following
- Benchmarking future data augmentation techniques

Technical Contributions

- Real-time data collection script, saved many hours of labeling data
- Real-time streaming of convolutional layer outputs
- Packaged helper functions and analysis methods in Python module, contributing to RRAV repository



Confidence: 0.67219561
(Predicted, Actual): (L, R)

Confidence: 0.99999988
(Predicted, Actual): (R, R)

Feature maps from a model trained on simulated data with shot noise applied. The top image is the preprocessed, raw image; the middle and bottom images are the channel-averaged outputs from the first and second convolutional layers.

Acknowledgements

Rory Lipkis, Adrian Agogino, Ignacio López-Francos, Pavlo Vlastos, the RRAV group, RSE & Intelligent Systems Division, N-269 summer 2023 interns, and everyone I was fortunate enough to meet during my internship.

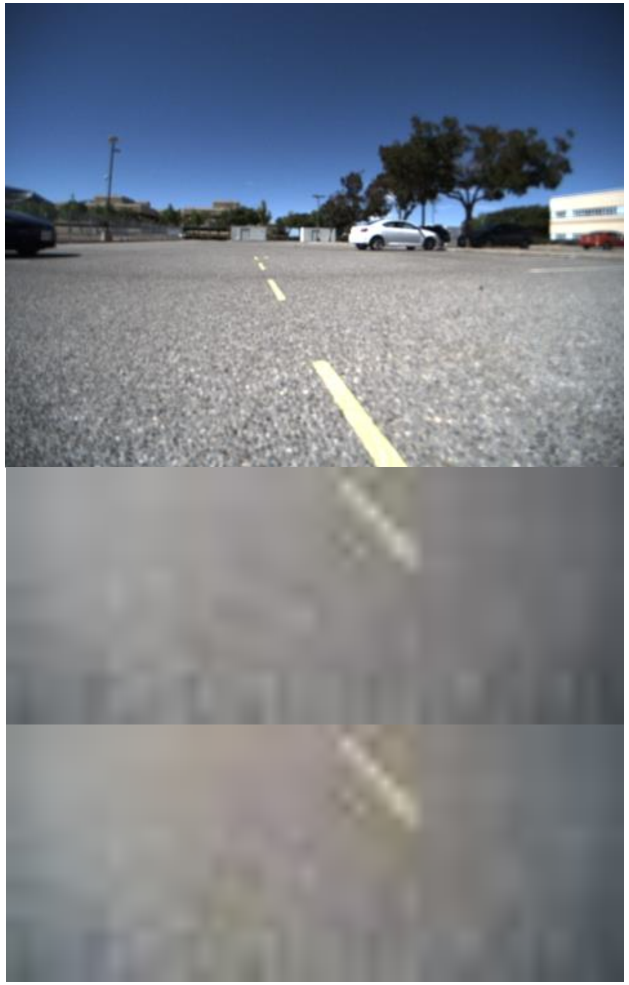
This work was conducted during an Office of STEM Engagement undergraduate internship and was supported by the System-Wide Safety (SWS) Project under the NASA Aeronautics Research Mission Directorate (ARMD) Airspace Operations and Safety Program (AOSP).

Thank you!

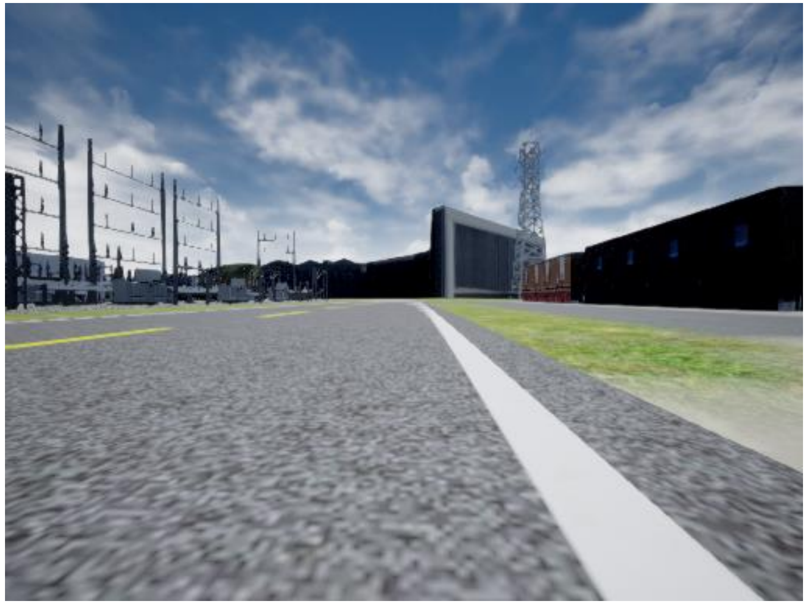
Contact me:
sfluo@berkeley.edu
stevenfluo.github.io

Initial Results

- Bad performance when models trained on simulated data were evaluated on real dataset
  - Brightness, line somewhat resembles a faded yellow or white line
- Models trained on the `shot_sim` dataset had sim2real accuracy of 61.46% (10 random seeds)
- Solution: saturate real images in real dataset



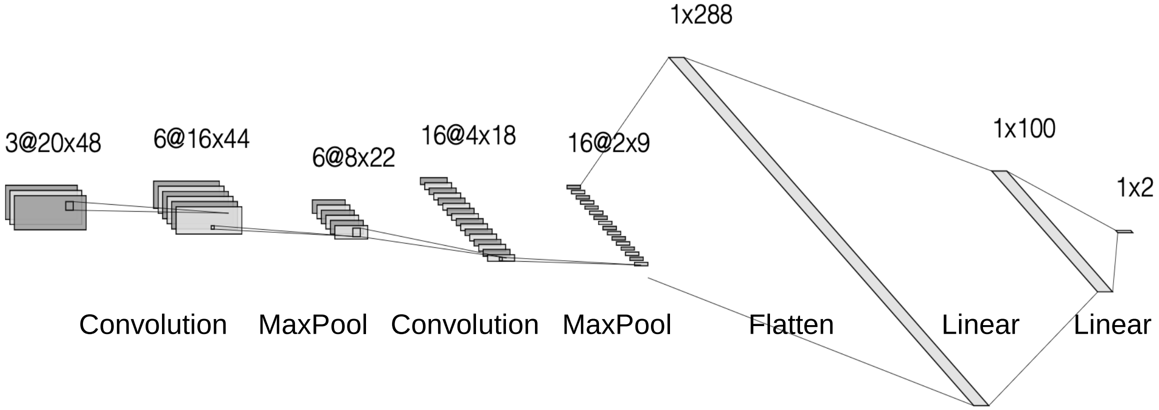
Dataset Limitations



Model Information: Left/Right Classifier

```python
class ConvNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)   # 5x5, 6 channels
        self.maxpool = nn.MaxPool2d(2, 2) # 2x2 max pool
        self.conv2 = nn.Conv2d(6, 16, 5)  # 5x5, 16 channels
        self.lin1 = nn.Linear(288, 100)   # fully connected to hidden layer
        self.lin2 = nn.Linear(100, 2)     # fully connected to logit output

    def forward(self, x):
        x = self.maxpool(F.leaky_relu(self.conv1(x)))
        x = self.maxpool(F.leaky_relu(self.conv2(x)))
        x = torch.flatten(x, 1)
        x = self.lin2(F.leaky_relu(self.lin1(x)))
        return x
```



Hyperparameters
Batch size: 128
Learning rate: 1e-3
Epochs: 25