

# BEAM



**WIRTSCHAFTS  
UNIVERSITÄT  
WIEN VIENNA  
UNIVERSITY OF  
ECONOMICS  
AND BUSINESS**

Boxology`s Extension for Annotating Machine Learning Systems



# Agenda

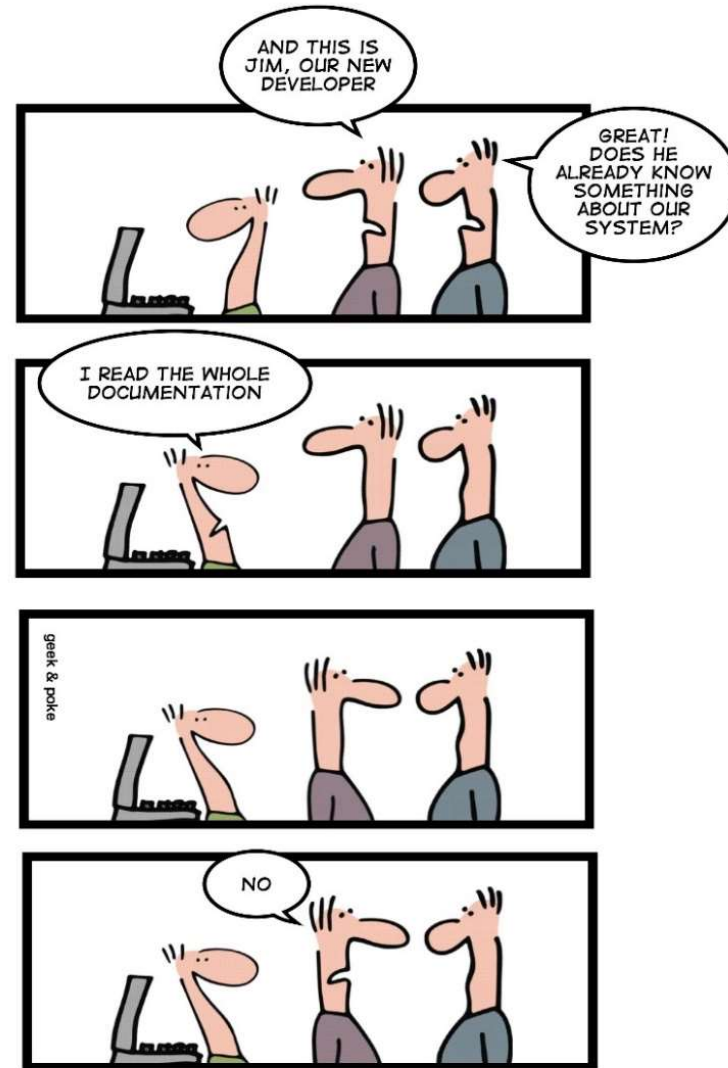
- Why Bother?
- Elements
- Element Properties
- Rules
- Q & A

*(If interest):*

- Known Limitations
- Future Work

# Why Bother?

- Diagrams as communication tool
- Understandable to most readers



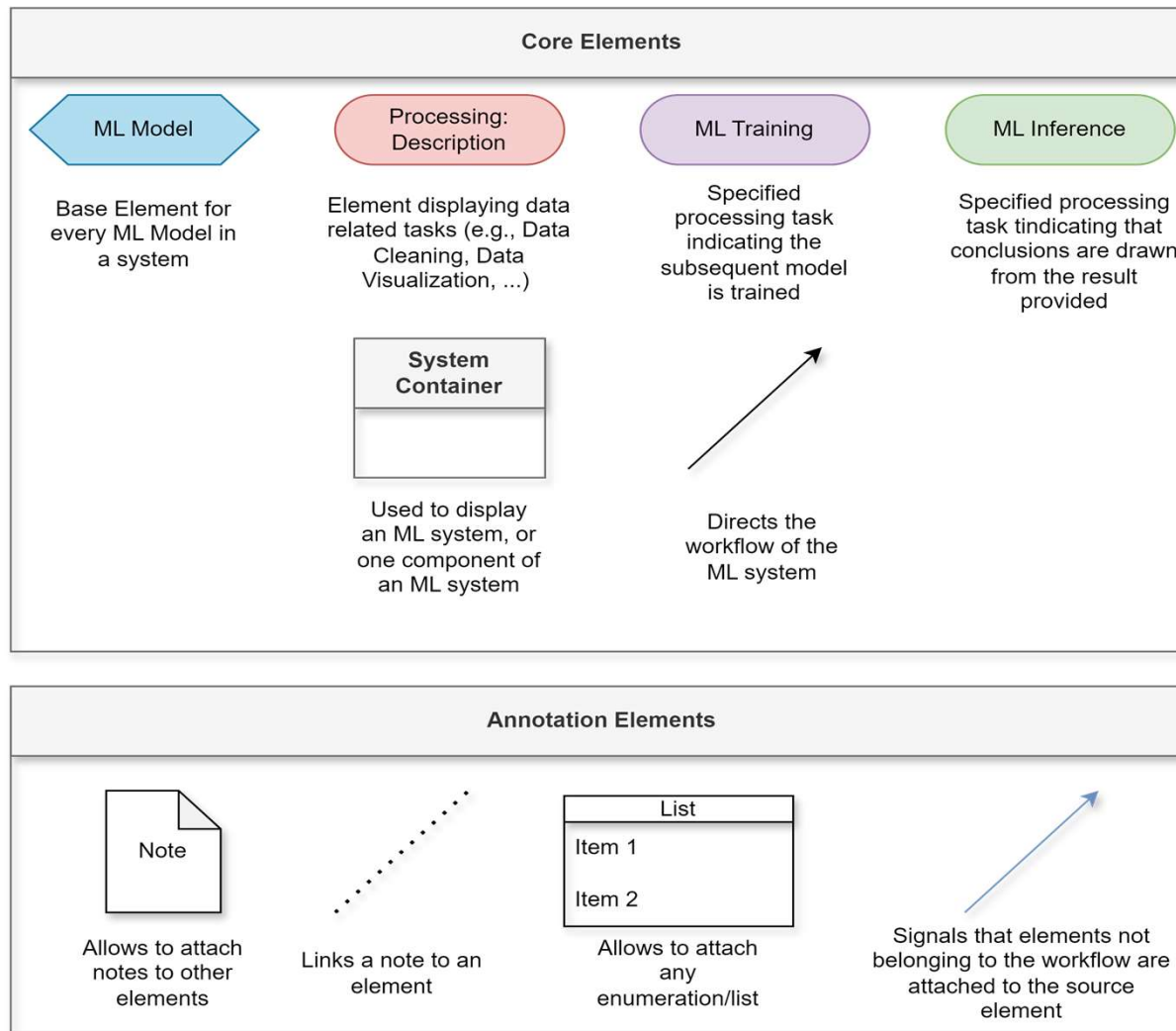
# On Documenting ML Models

```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10)
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits

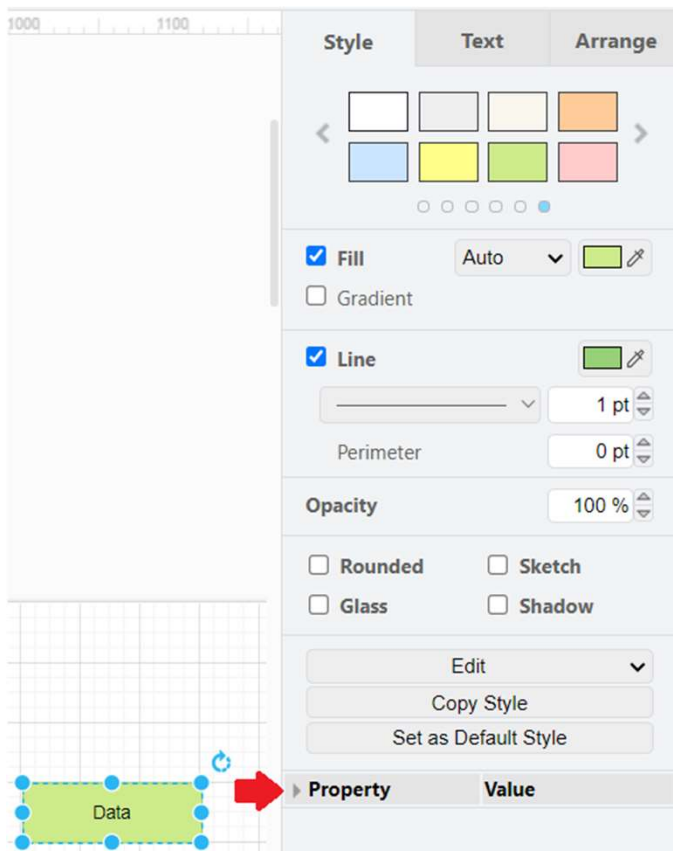
model = NeuralNetwork().to(device)
print(model)
```

# Elements

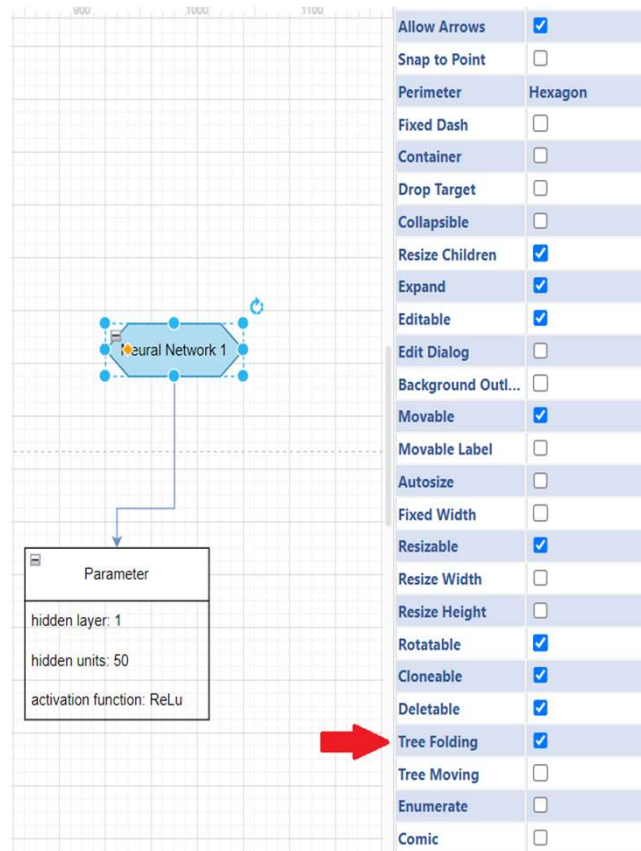


# Element Properties

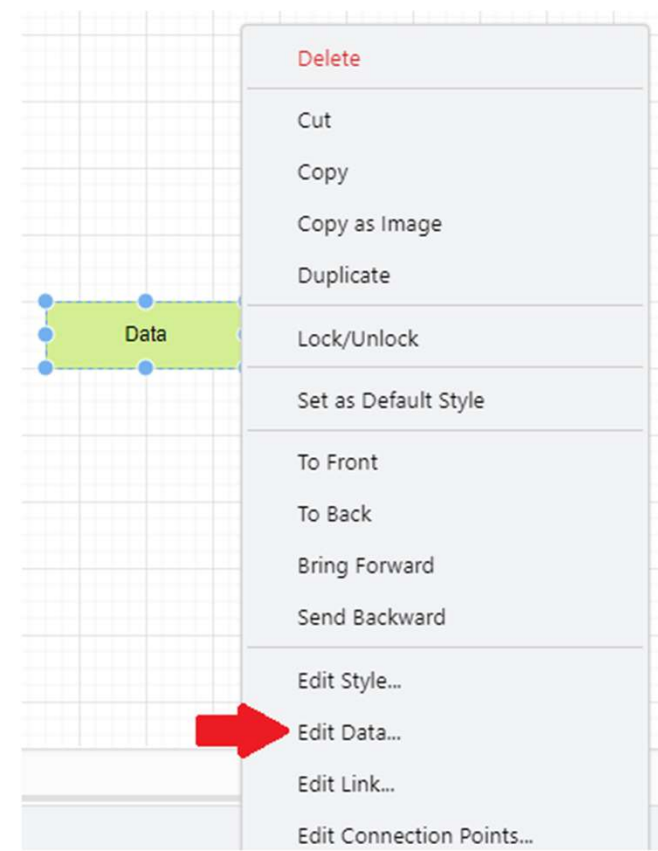
## Open default properties



## Select default properties



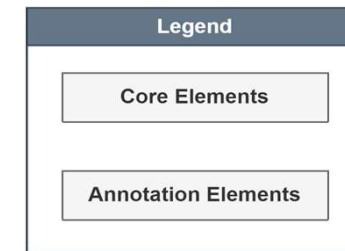
## Assign custom properties



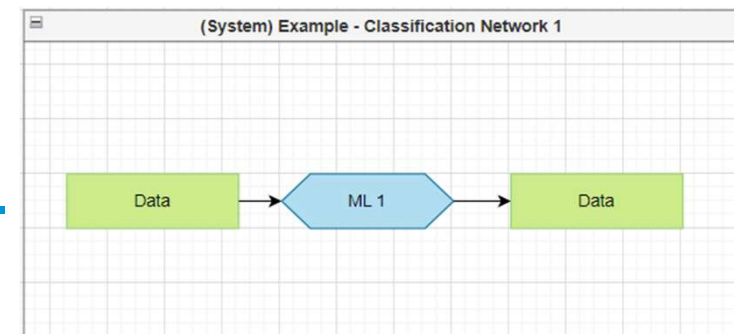
# Rules

1. Legend
2. At least one system
3. At least one workflow
4. "element" property assigned
5. Newly introduced elements have to be added to the legend

1.



2. & 3.

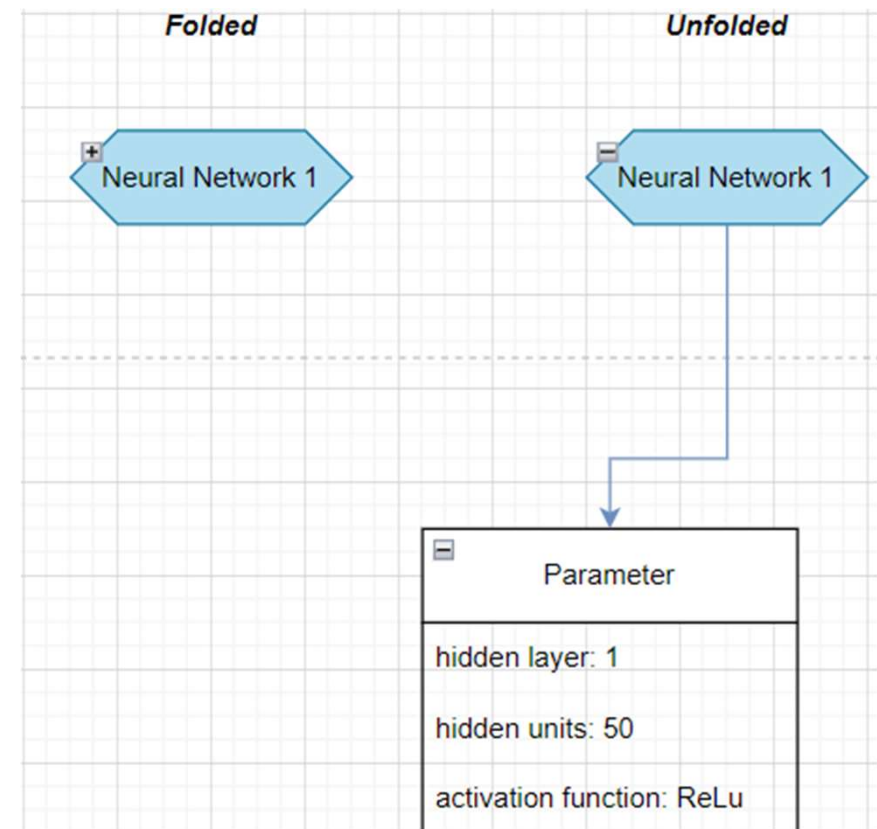


4.



# Attaching Further Details

- Own connector to distinguish
- Relevant for interactive diagrams & code in the background



# Q & A



# Backup



# Known Limitations

- Flexibility vs. Robustness
  - Room for ambiguity despite legend
- Annotating large quantities of details (e.g., Properties)
- Input Restrictions for elements
- Introducing semantic rules

# Future Work

- For Draw.io
  - Different Layers
  - Intern Links
- Autonomous code extraction
- Diagram generation via code + stored building elements
- Introducing more granular syntactic & semantic rules, which do not take away too much flexibility