VIS 2025

Doctoral Colloquium

This Slide

# Towards a Generalized Deep Learning Visual Analytics Design

Rahat Zaman

PhD Student

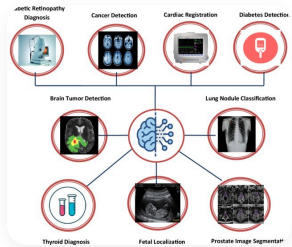
Advisor: Paul Rosen

SCI Institute · University of Utah

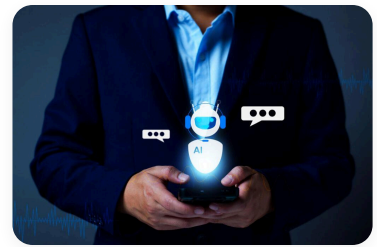
# Deep Learning




Finance




Healthcare

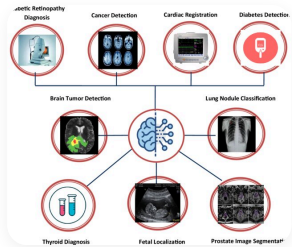



Personal Assistant

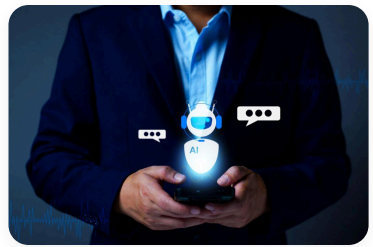
# Deep Learning




Finance




Healthcare



Personal Assistant

## Various Deep Learning Frameworks

**Programming**

PyTorch

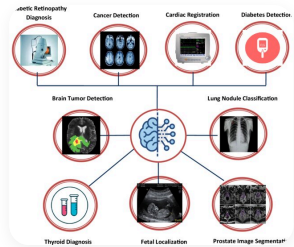
TensorFlow

JAX

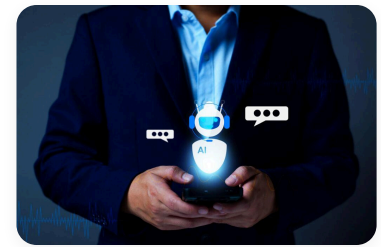
# Deep Learning




Finance




Healthcare



Personal Assistant

## Various Deep Learning Frameworks

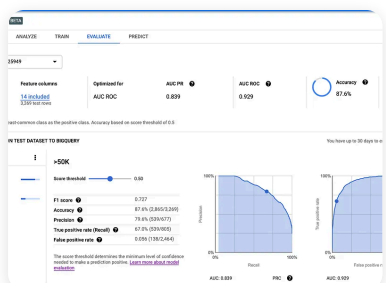
### Programming

PyTorch

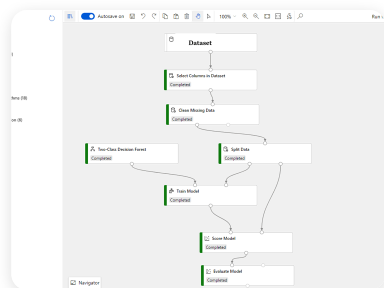
TensorFlow

JAX

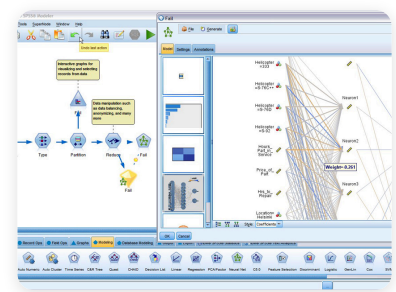
### Visual Model Builders



Google Cloud AutoML

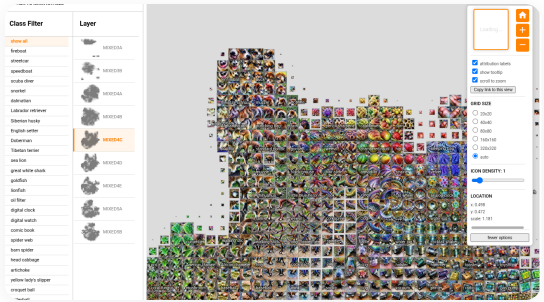



Azure Visual Builder



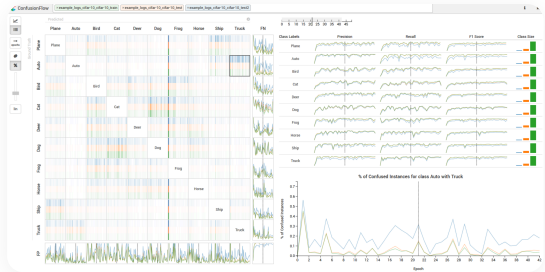
IBM SPSS Modeler

# Beyond Building: Model Visualization



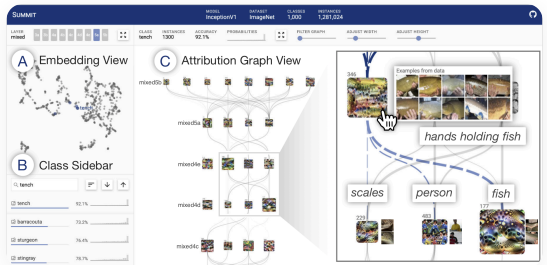

**Activation Atlas**

Dimension Reduction of Activation



**Confusion Flow**

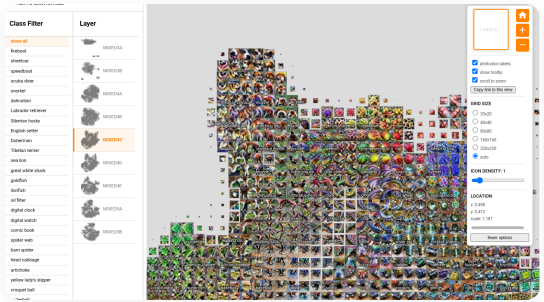
Visualizes Confusion Matrix




**Summit**

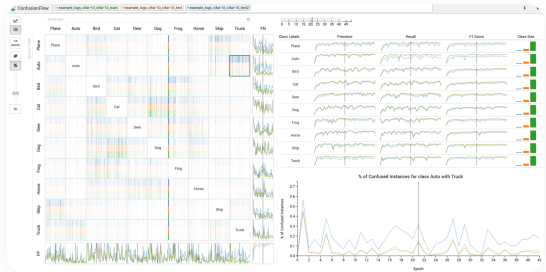
Visualizes Activation Channels

# Beyond Building: Model Visualization



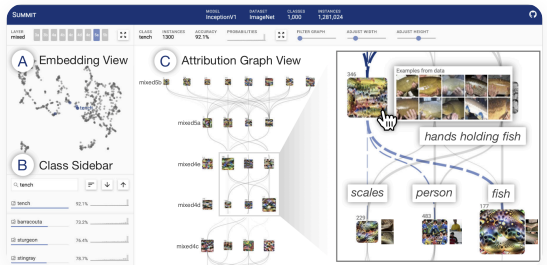

**Activation Atlas**

Dimension Reduction of Activation



**Confusion Flow**

Visualizes Confusion Matrix

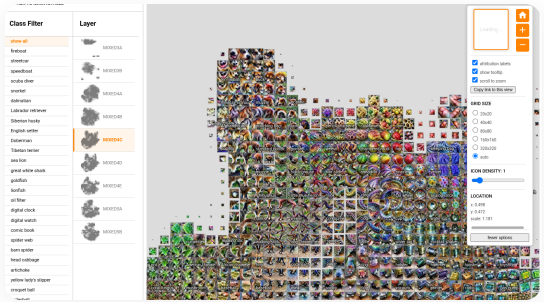



**Summit**

Visualizes Activation Channels

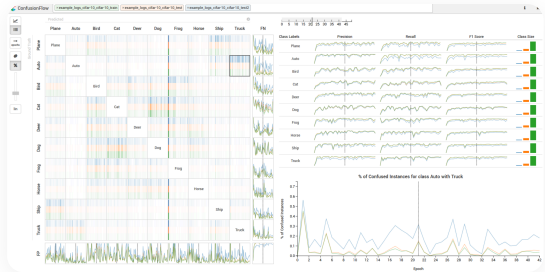
- Improves Model Transparency
- Helps in Debugging
- Helps in Trusting the Model

# Beyond Building: Model Visualization



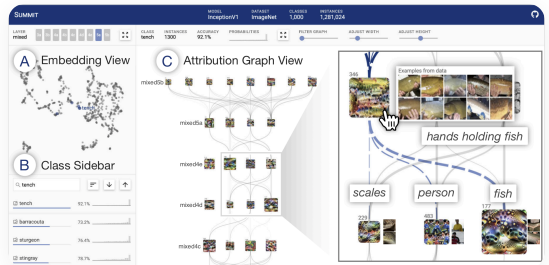

**Activation Atlas**

Dimension Reduction of Activation



**Confusion Flow**

Visualizes Confusion Matrix




**Summit**

Visualizes Activation Channels

- Improves Model Transparency
- Helps in Debugging
- Helps in Trusting the Model

## Research Gap

- Computational and Visual Scalability vs Generalization
- No Explorable Visualizations
- Low Transferability & Requires Extensive Programming Effort

**Deep Learning Visual Analytics should be generalized, accessible, explorable and collaborative, so it can be used for both explaining and debugging.**

Deep Learning Visual Analytics should be generalized, accessible, explorable and collaborative, so it can be used for both explaining and debugging.

# Dissertation Goal

### Generalized Visualization

Create a flexible visualization method for any neural networks.

### Exploratory Model Visualization

Enable dynamic visualization that can be used to explore/debug/verify the model.

### Low Programming Effort

Reduce the programming effort required to create visualizations for new models.

Deep Learning Visual Analytics should be generalized, accessible, explorable and collaborative, so it can be used for both explaining and debugging.

# Dissertation Goal

### Generalized Visualization

Create a flexible visualization method for any neural networks.

### Exploratory Model Visualization

Enable dynamic visualization that can be used to explore/debug/verify the model.

### Low Programming Effort

Reduce the programming effort required to create visualizations for new models.

Deep Learning Visual Analytics should be generalized, accessible, explorable and collaborative, so it can be used for both explaining and debugging.

# Dissertation Goal

### Generalized Visualization

Create a flexible visualization method for any neural networks.

### Exploratory Model Visualization

Enable dynamic visualization that can be used to explore/debug/verify the model.

### Low Programming Effort

Reduce the programming effort required to create visualizations for new models.

Deep Learning Visual Analytics should be generalized, accessible, explorable and collaborative, so it can be used for both explaining and debugging.

# Dissertation Goal

### Generalized Visualization

Create a flexible visualization method for any neural networks.

### Exploratory Model Visualization

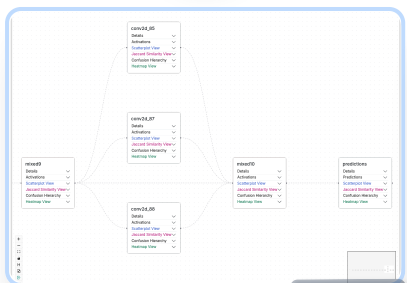
Enable dynamic visualization that can be used to explore/debug/verify the model.

### Low Programming Effort

Reduce the programming effort required to create visualizations for new models.

# PhD Milestones

1



Under Revision

## Design Special Analytics Tool

Identify Challenges

2

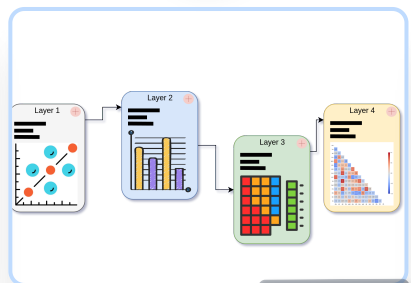



Under Revision

## Usage Scenarios & User Study

What works best

3

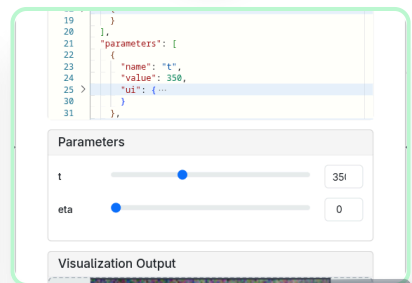



Planned VIS'26/AAAI

## Generalized DL Vis Model

Computationally & Visually Scalable

4



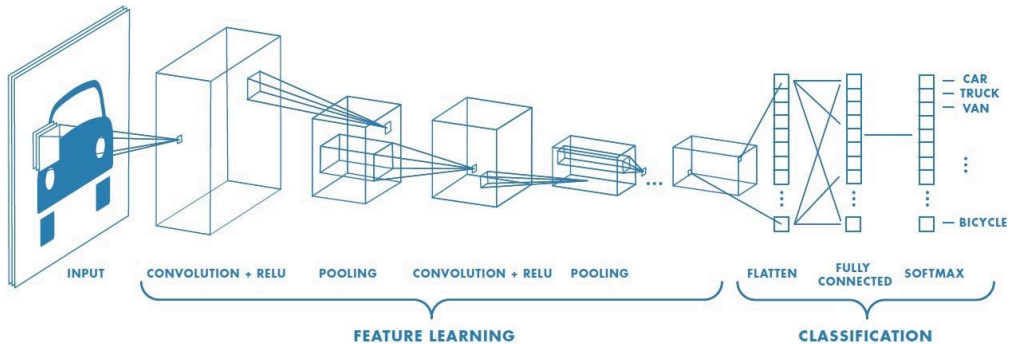

Planned VIS'26

## Declarative Layer Vis Grammar

Low Programming and Reproducible

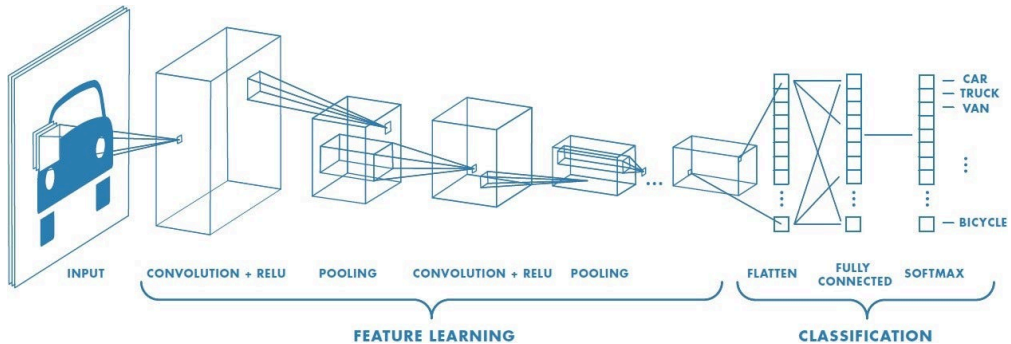
# Visualization Models with Image-based Layers

Layers that process spatial data like images, videos, etc.



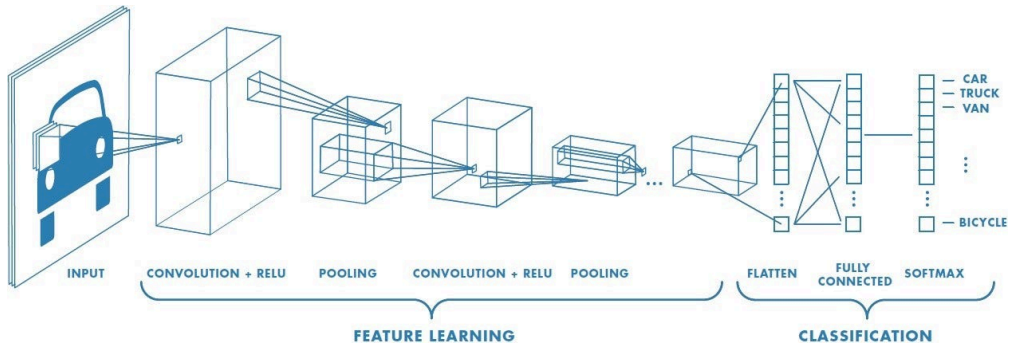
# Visualization Models with Image-based Layers

Layers that process spatial data like images, videos, etc.



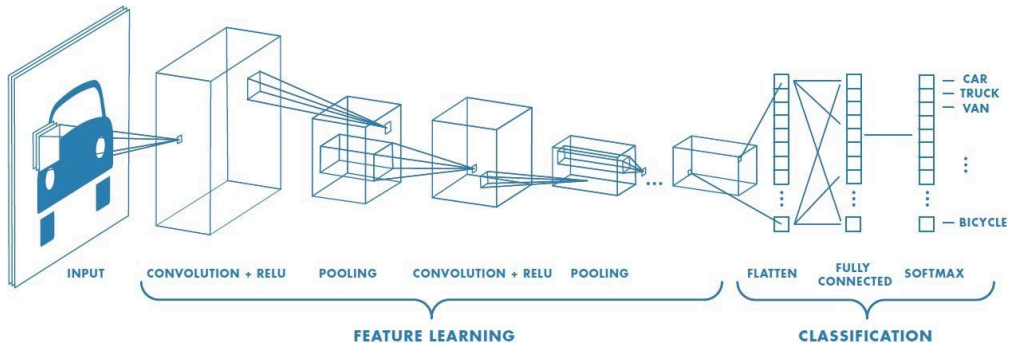
# Visualization Models with Image-based Layers

Layers that process spatial data like images, videos, etc.



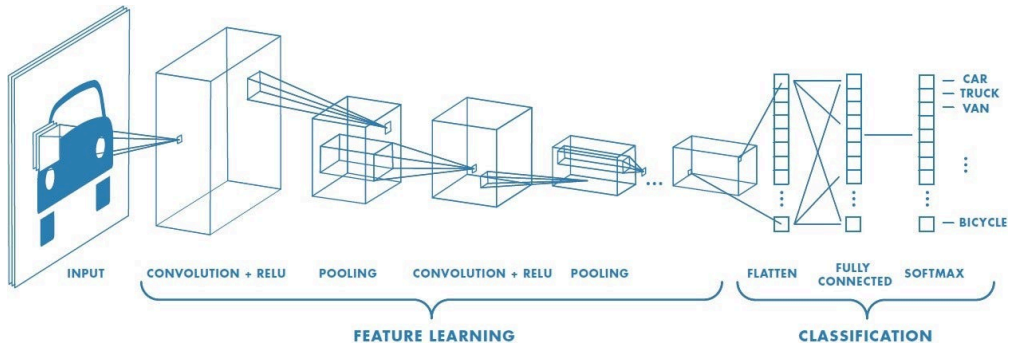
# Visualization Models with Image-based Layers

Layers that process spatial data like images, videos, etc.



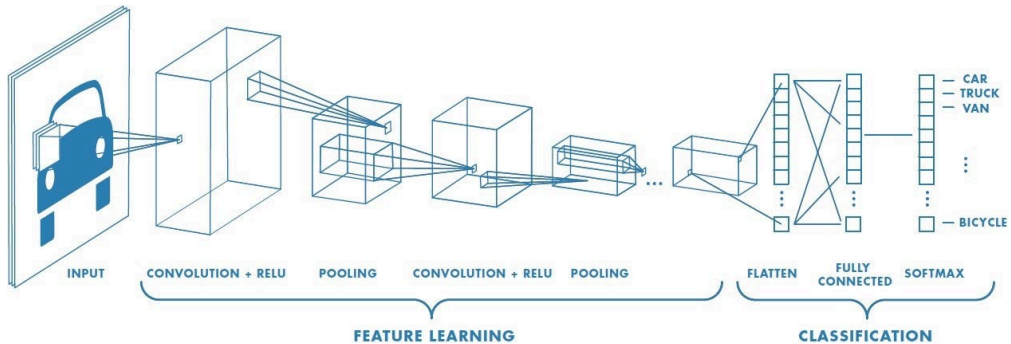
# Visualization Models with Image-based Layers

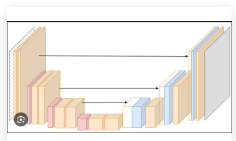
Layers that process spatial data like images, videos, etc.



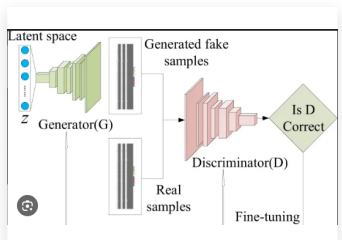
# Visualization Models with Image-based Layers

Layers that process spatial data like images, videos, etc.

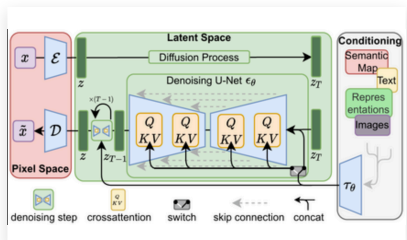


U-Net



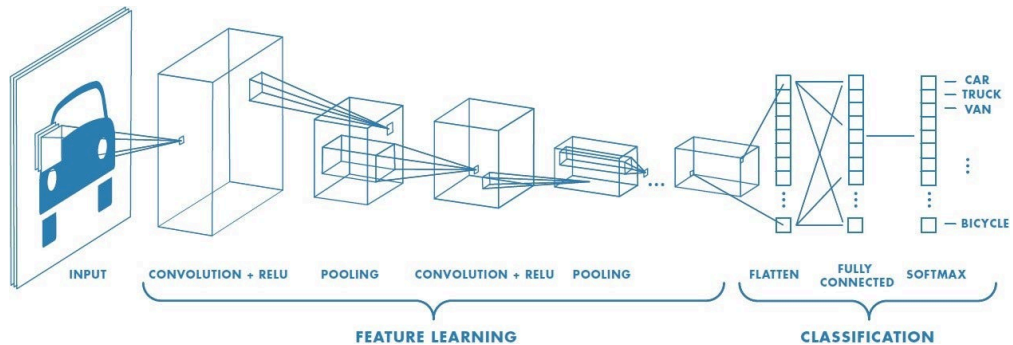

GAN




Stable Diffusion

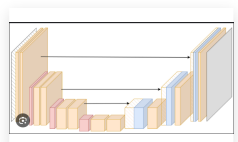
# Visualization Models with Image-based Layers

Layers that process spatial data like images, videos, etc.

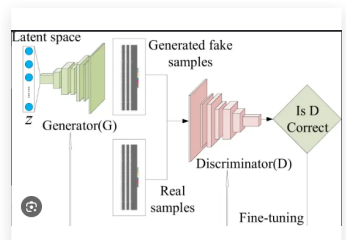



## Challenges

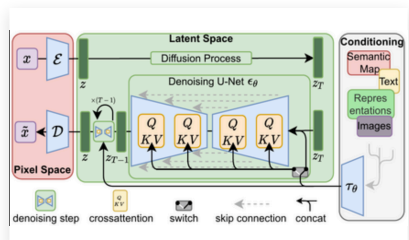
- High Dimensionality
- Multiple Layers and Depth
- Scalability of Visualizations



U-Net

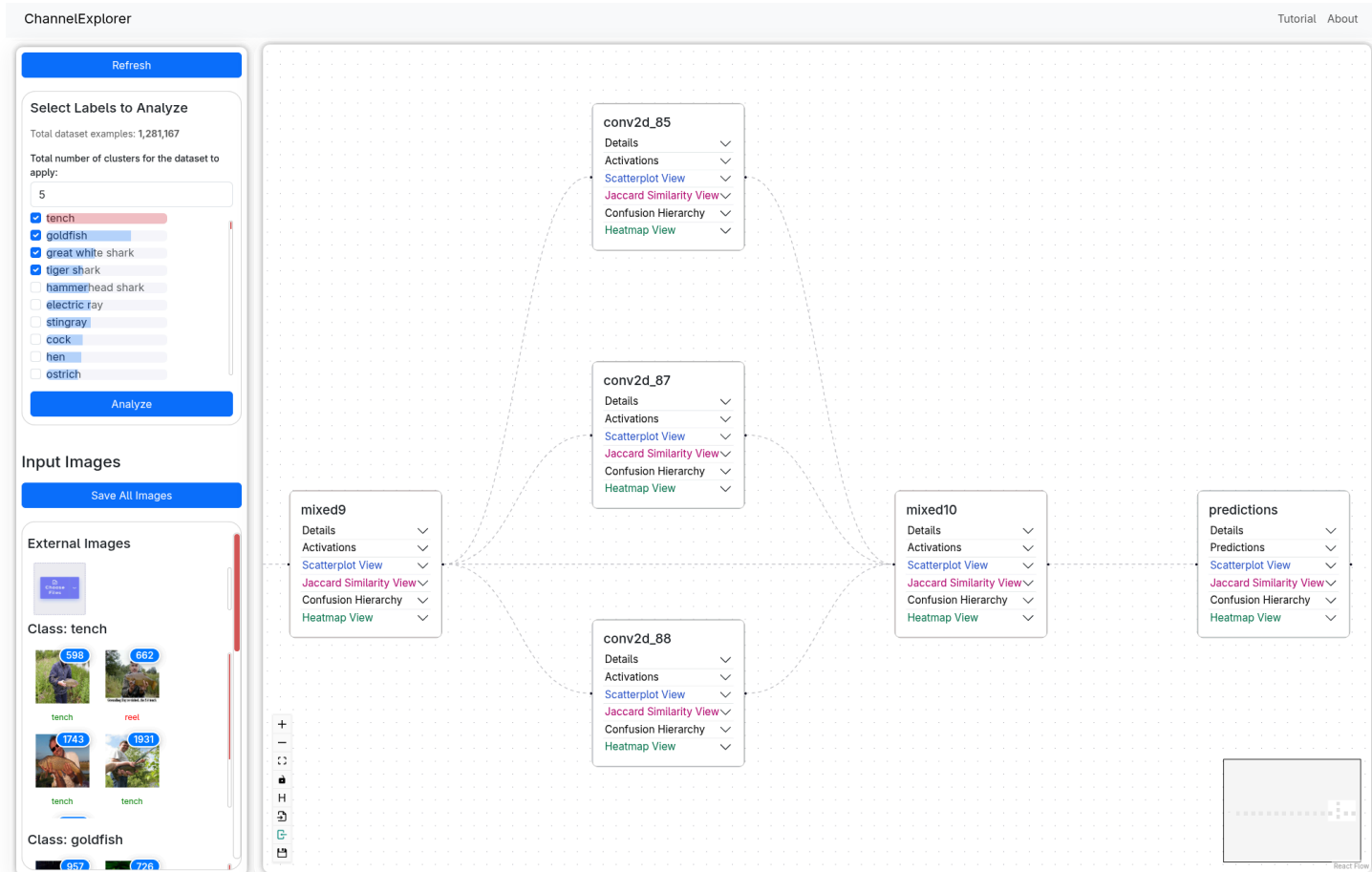



GAN

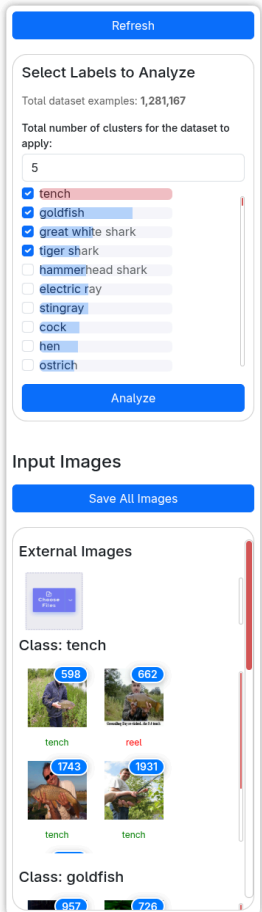



Stable Diffusion

# ChannelExplorer

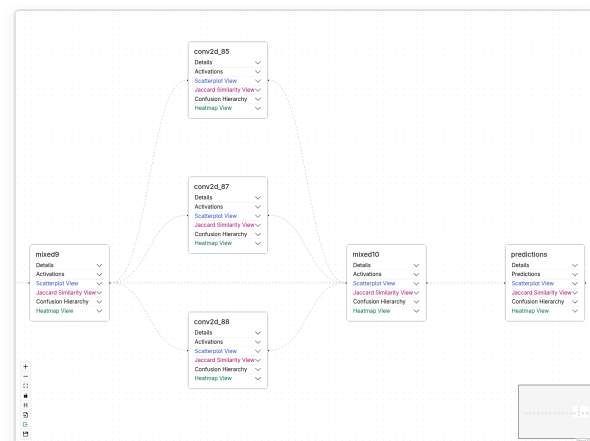


# ChannelExplorer

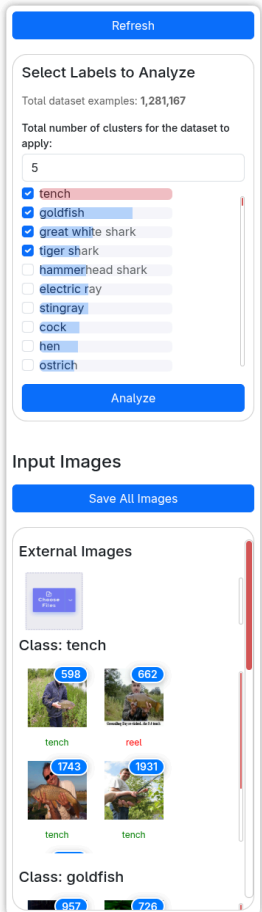



## Dataset View

- group-wide **activation distances**
- Selection of **Representative Inputs**
- How many inputs represented by each Representative Input
- Upload custom inputs



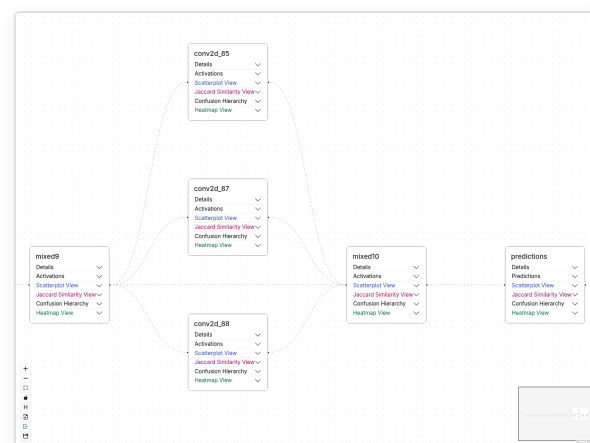
# ChannelExplorer



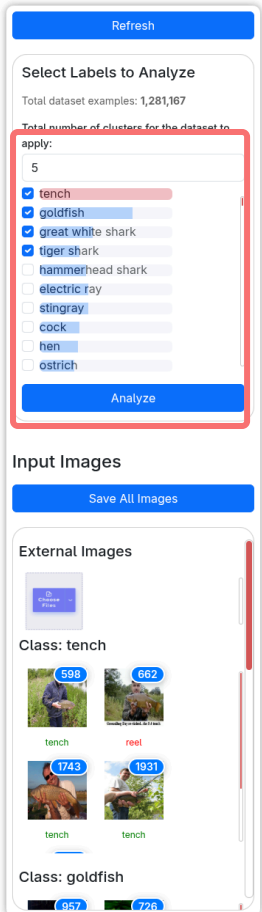

## Dataset View

- group-wide **activation distances**
- Selection of **Representative Inputs**
- How many inputs represented by each Representative Input
- Upload custom inputs

Class 1 Class 2 Class 3



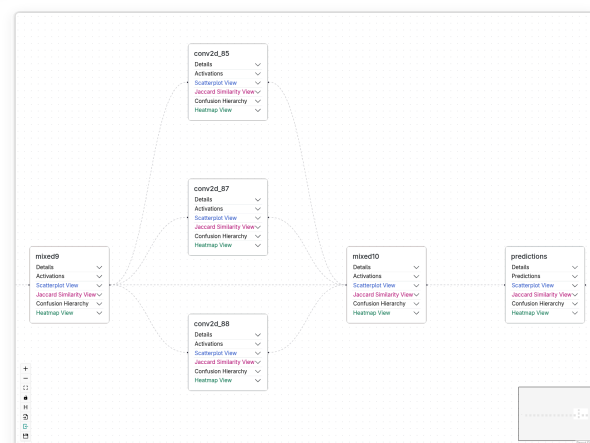
# ChannelExplorer



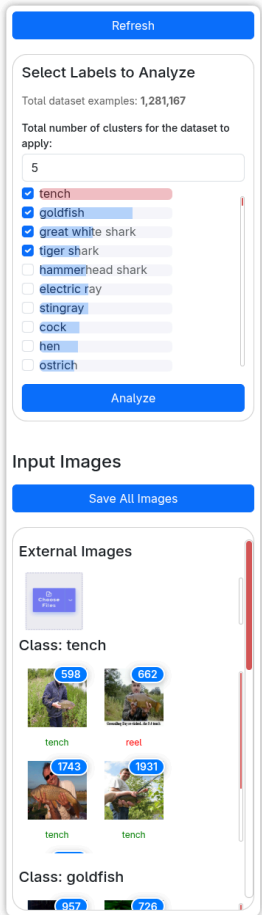

## Dataset View

- group-wide **activation distances**
- Selection of **Representative Inputs**
- How many inputs represented by each Representative Input
- Upload custom inputs

Class 1 Class 2 Class 3

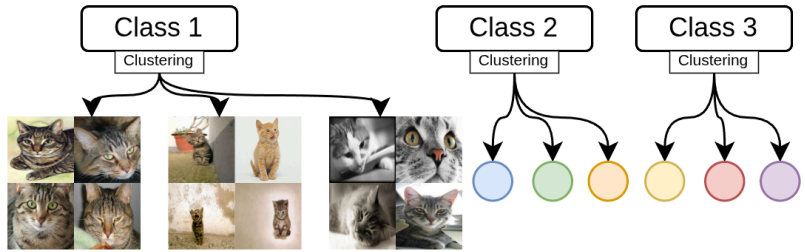


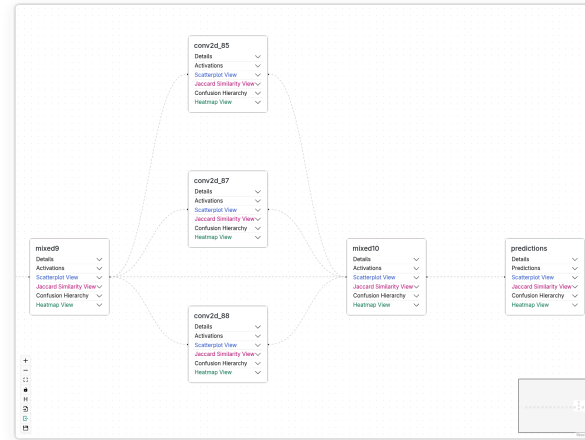
# ChannelExplorer



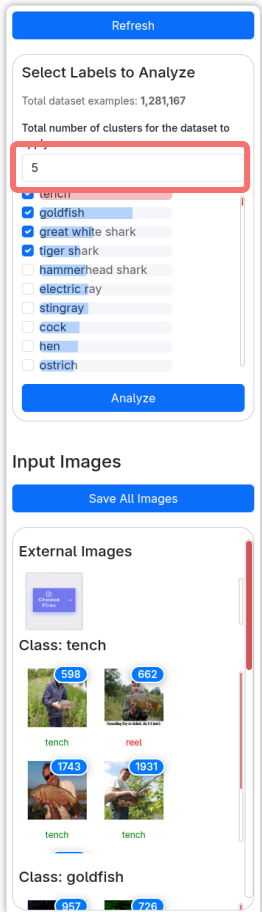

## Dataset View

- group-wide **activation distances**
- Selection of **Representative Inputs**
- How many inputs represented by each Representative Input
- Upload custom inputs



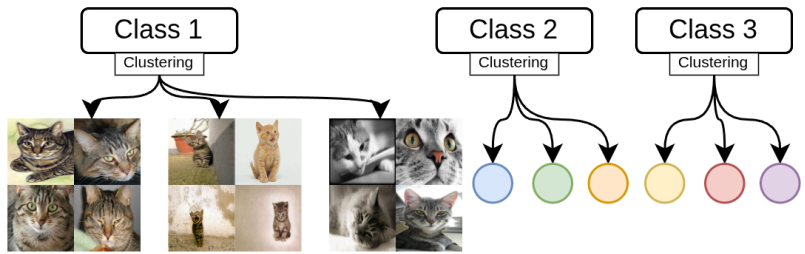



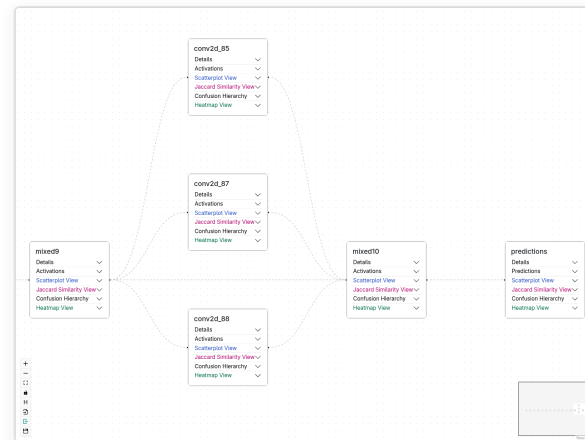
# ChannelExplorer



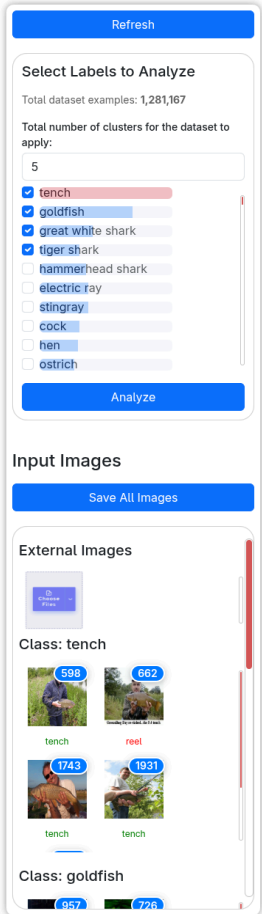

## Dataset View

- group-wide **activation distances**
- Selection of **Representative Inputs**
- How many inputs represented by each Representative Input
- Upload custom inputs



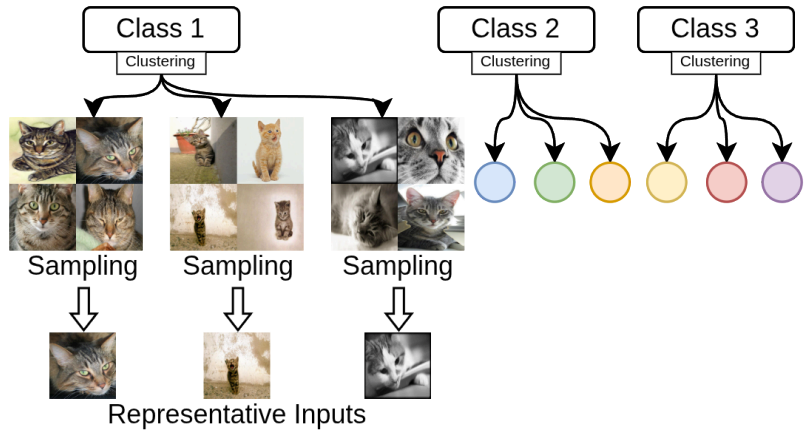



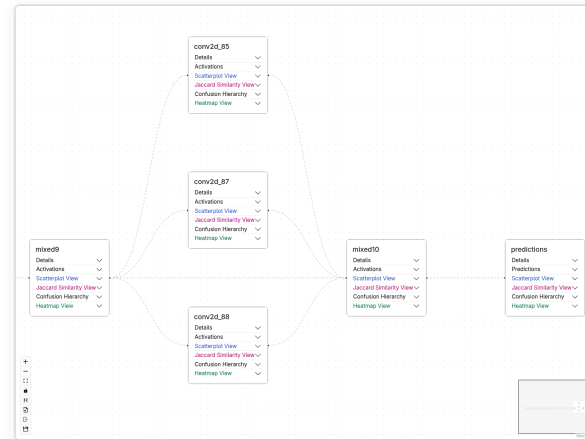
# ChannelExplorer



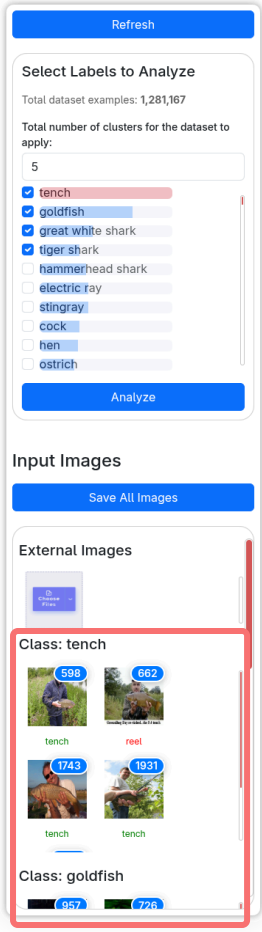

## Dataset View

- group-wide **activation distances**
- Selection of **Representative Inputs**
- How many inputs represented by each Representative Input
- Upload custom inputs



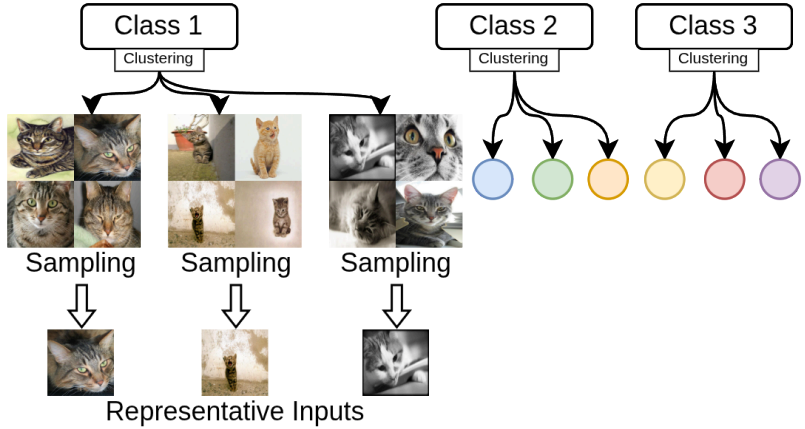



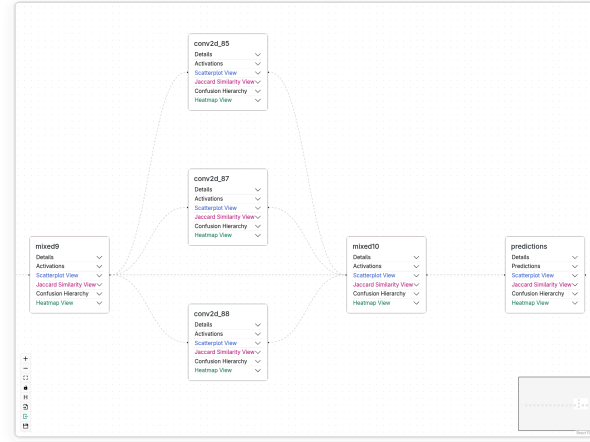
# ChannelExplorer



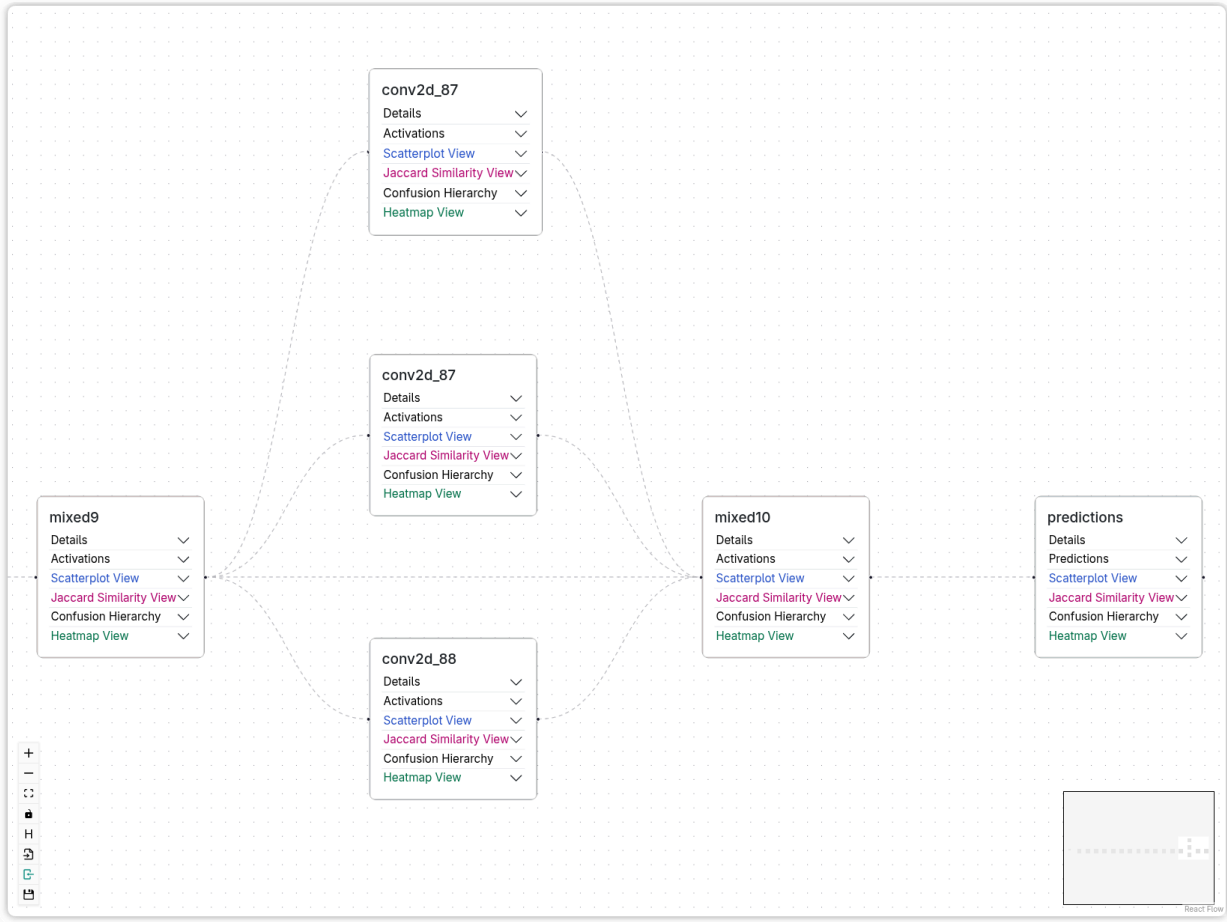

## Dataset View

- group-wide **activation distances**
- Selection of **Representative Inputs**
- How many inputs represented by each Representative Input
- Upload custom inputs



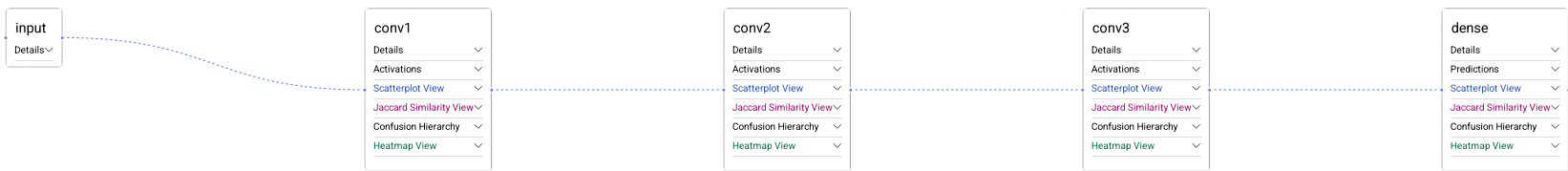
# ChannelExplorer



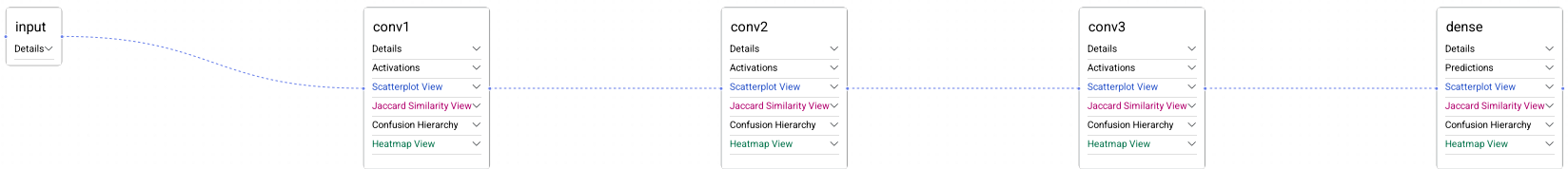

## Model View

- A Node-Link Diagram
- Nodes: Layers
- Edges: Connections between layers

# Node-Link Diagram Demonstration

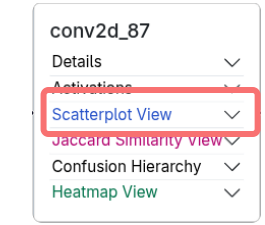


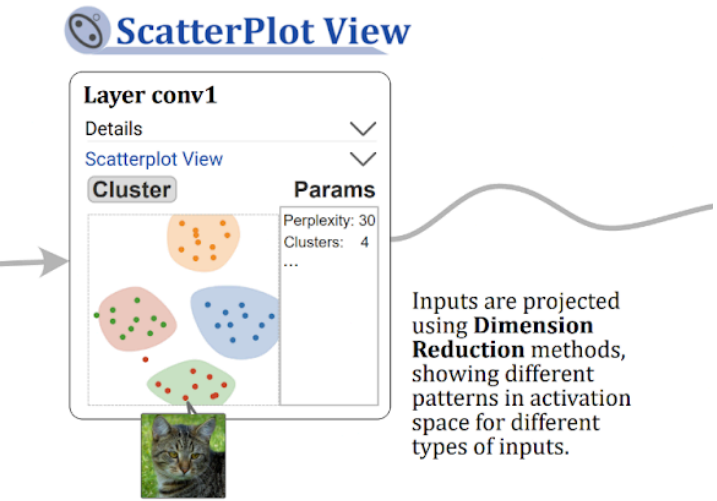
# Node-Link Diagram Demonstration




Live Demo: [l.insane.casa/channelexplorer](l.insane.casa/channelexplorer)

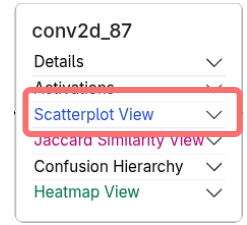
# ChannelExplorer (Three Exploration Views)

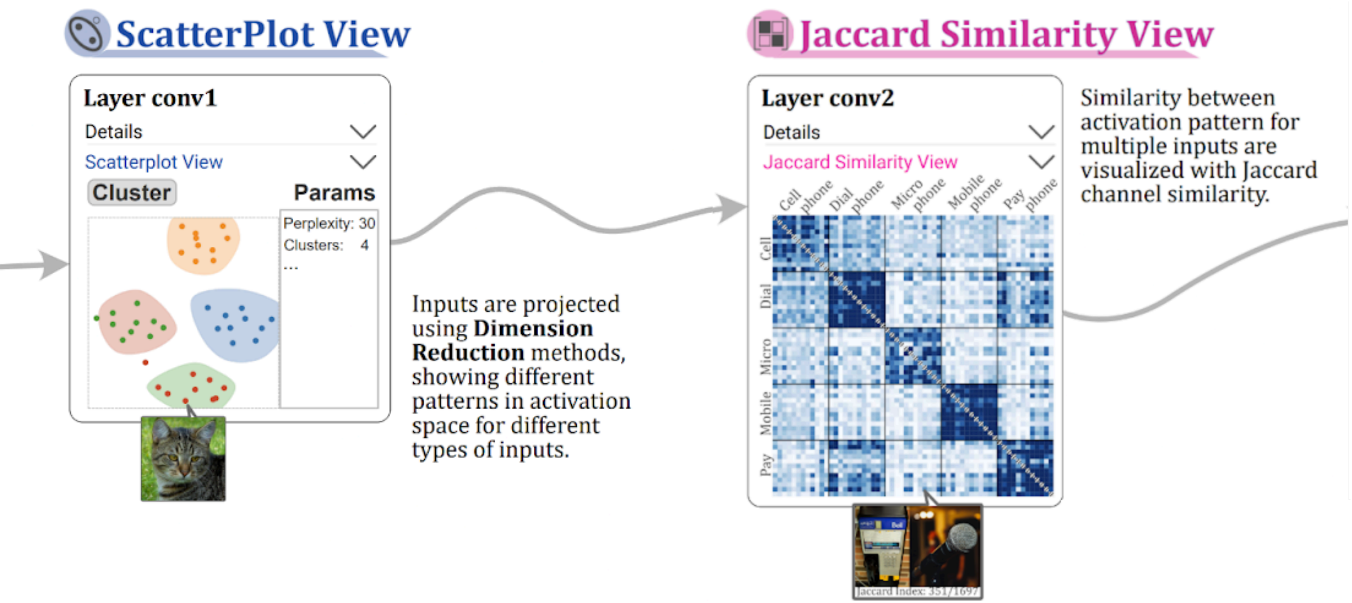


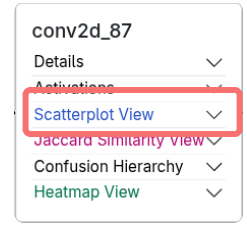

Inputs are projected using **Dimension Reduction** methods, showing different patterns in activation space for different types of inputs.

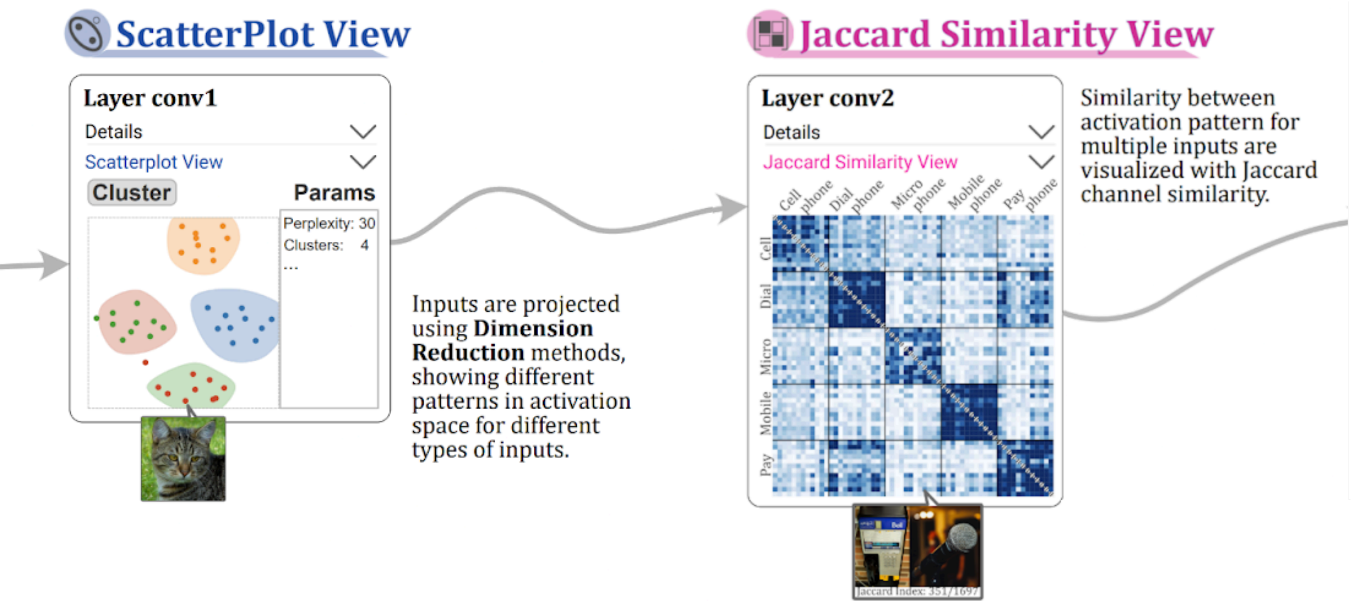
# ChannelExplorer (Three Exploration Views)



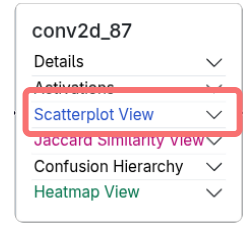



# ChannelExplorer (Three Exploration Views)

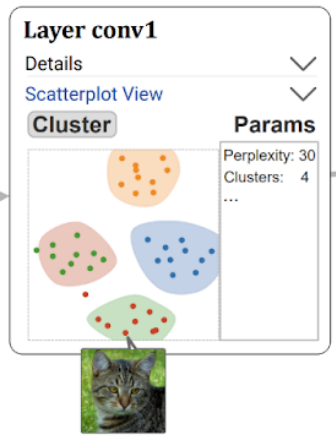


# ChannelExplorer (Three Exploration Views)

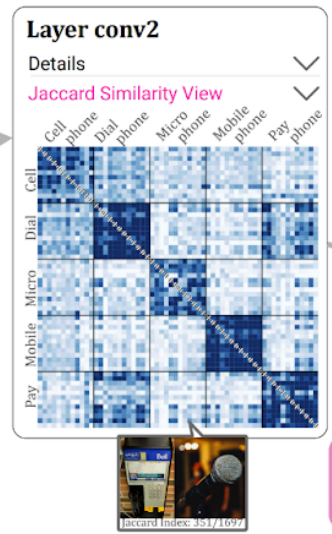



## ScatterPlot View



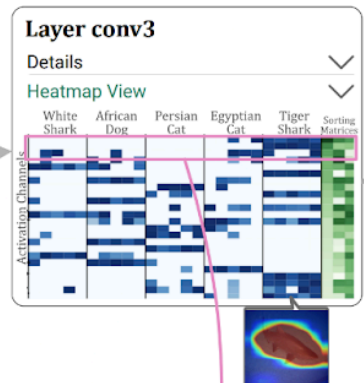

Inputs are projected using **Dimension Reduction** methods, showing different patterns in activation space for different types of inputs.

## Jaccard Similarity View




Similarity between activation pattern for multiple inputs are visualized with Jaccard channel similarity.

## Heatmap View

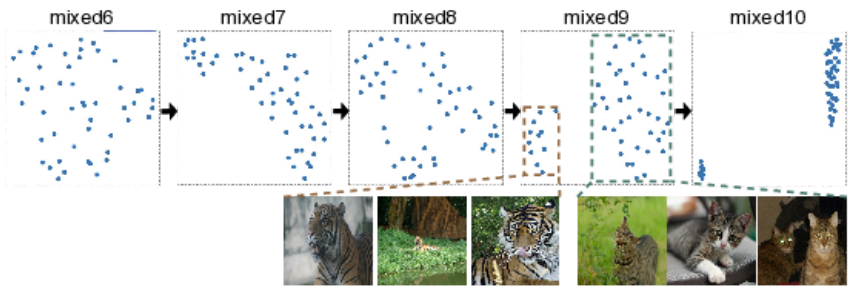



Individual channels for multiple inputs are sorted and visualized using heatmap. Bright stripes indicate filters targeting specific features.

# Applications of ChannelExplorer

## Mislabel Identification

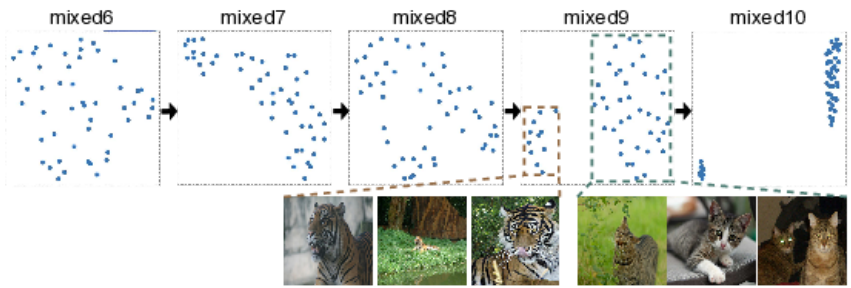
Scatterplot view of ChannelExplorer shows that a classification dataset have mislabeled inputs. In ImageNet, cats are labeled as Tiger Cat.



# Applications of ChannelExplorer

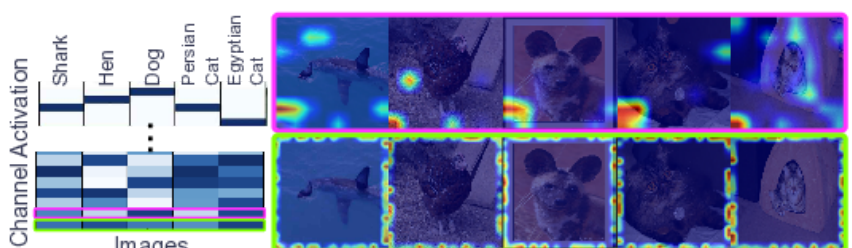
## Mislabel Identification

Scatterplot view of ChannelExplorer shows that a classification dataset have mislabeled inputs. In ImageNet, cats are labeled as Tiger Cat.




## Unimportant Channels Identification

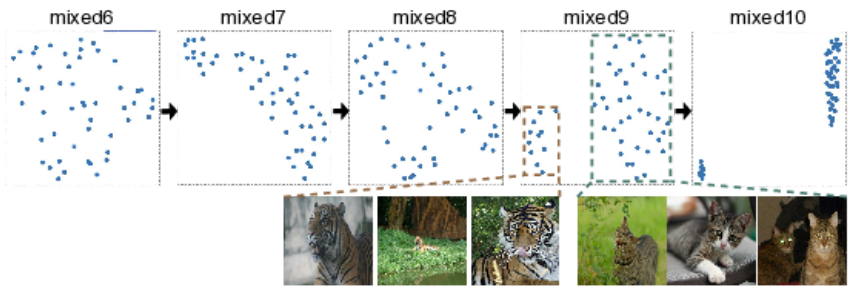
Channels activating for unimportant features - attending to borders of the image.



# Applications of ChannelExplorer

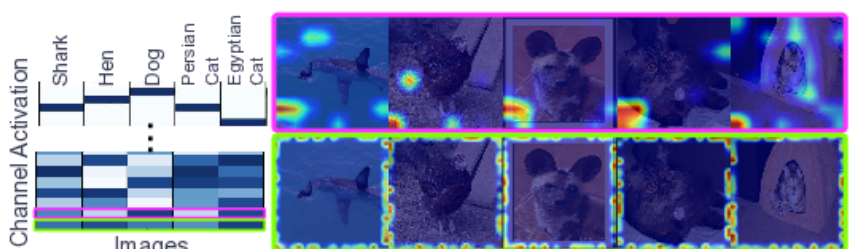
## Mislabel Identification

Scatterplot view of ChannelExplorer shows that a classification dataset have mislabeled inputs. In ImageNet, cats are labeled as Tiger Cat.



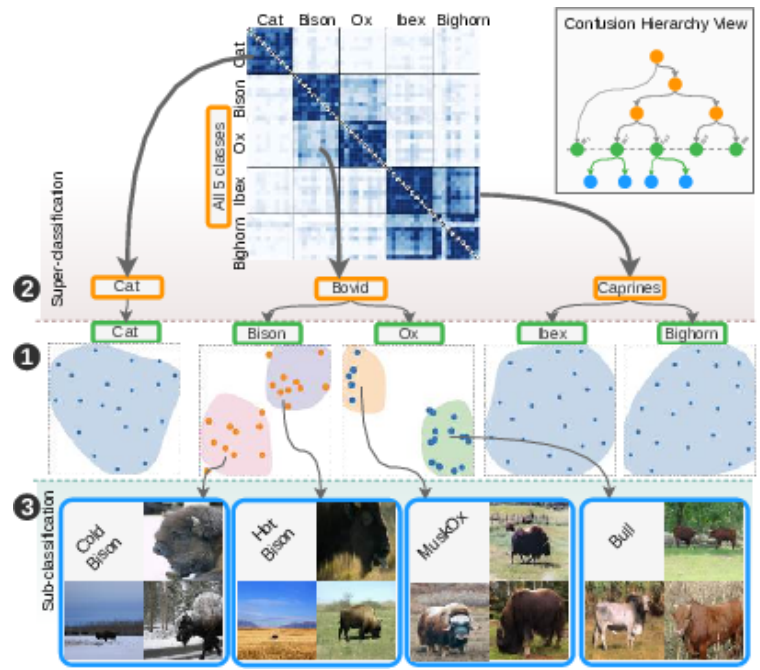

## Unimportant Channels Identification

Channels activating for unimportant features - attending to borders of the image.




## Confusion Hierarchical Generation with Reasoning

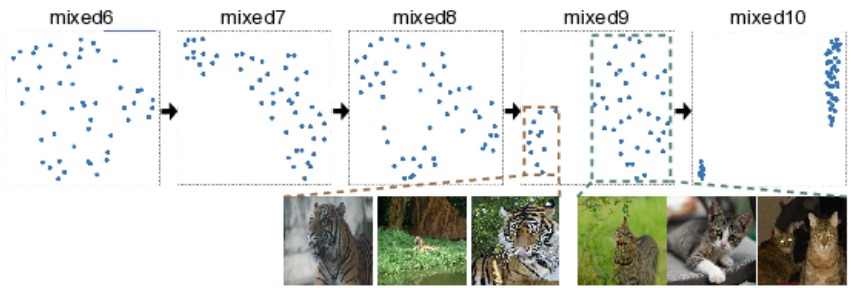
Starting from 5 classes, 3 super-classes (Cat, Bovid, and Caprine) are created. In the Bison's Scatterplot View, two clusters show examples of Bison in Cold & Hot weather. Similarly, Oxen's Scatterplot View shows furry and barrel-shaped bodies (MuskOx breed), and less furry cow-like bodies (Bull); creating 2 subclasses.



# Applications of ChannelExplorer

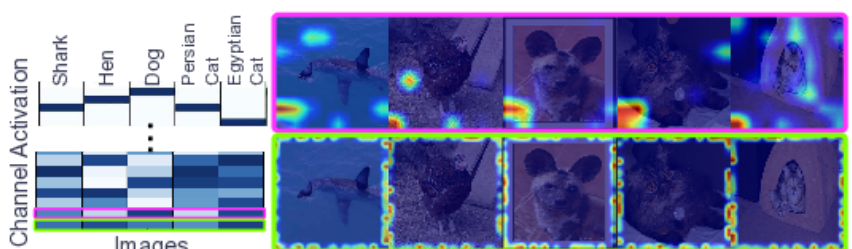
## Mislabel Identification

Scatterplot view of ChannelExplorer shows that a classification dataset have mislabeled inputs. In ImageNet, cats are labeled as Tiger Cat.



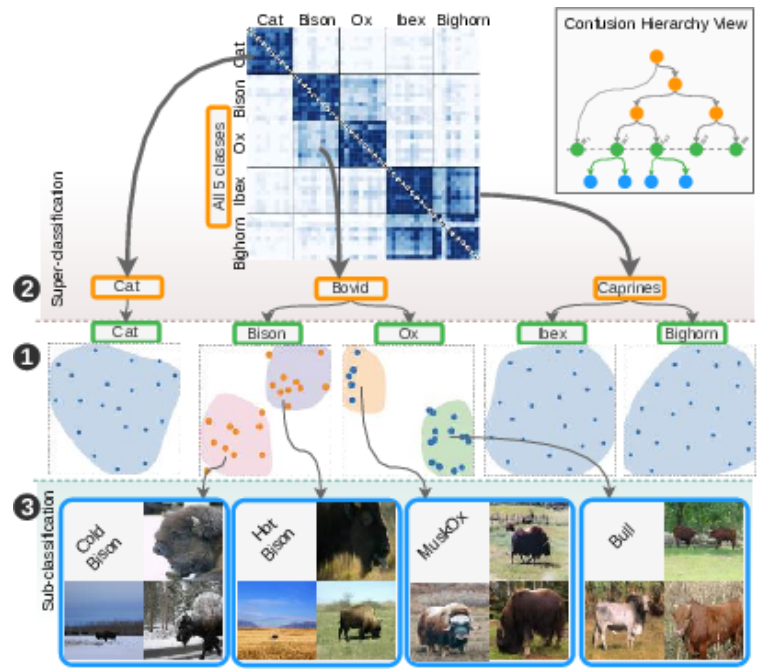

## Unimportant Channels Identification

Channels activating for unimportant features - attending to borders of the image.




## Confusion Hierarchical Generation with Reasoning

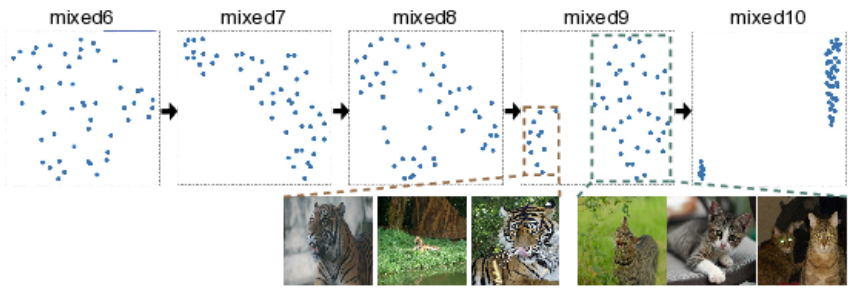
Starting from 5 classes, 3 super-classes (Cat, Bovid, and Caprine) are created. In the Bison's Scatterplot View, two clusters show examples of Bison in Cold & Hot weather. Similarly, Oxen's Scatterplot View shows furry and barrel-shaped bodies (MuskOx breed), and less furry cow-like bodies (Bull); creating 2 subclasses.



# Applications of ChannelExplorer

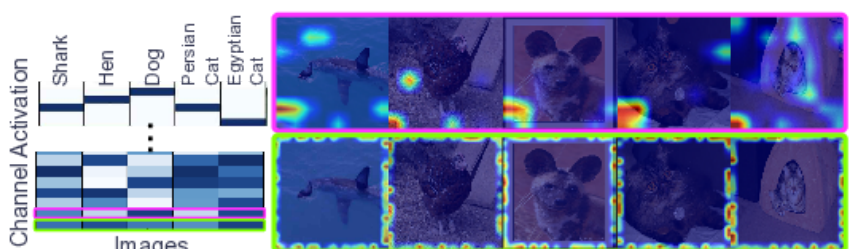
## Mislabel Identification

Scatterplot view of ChannelExplorer shows that a classification dataset have mislabeled inputs. In ImageNet, cats are labeled as Tiger Cat.



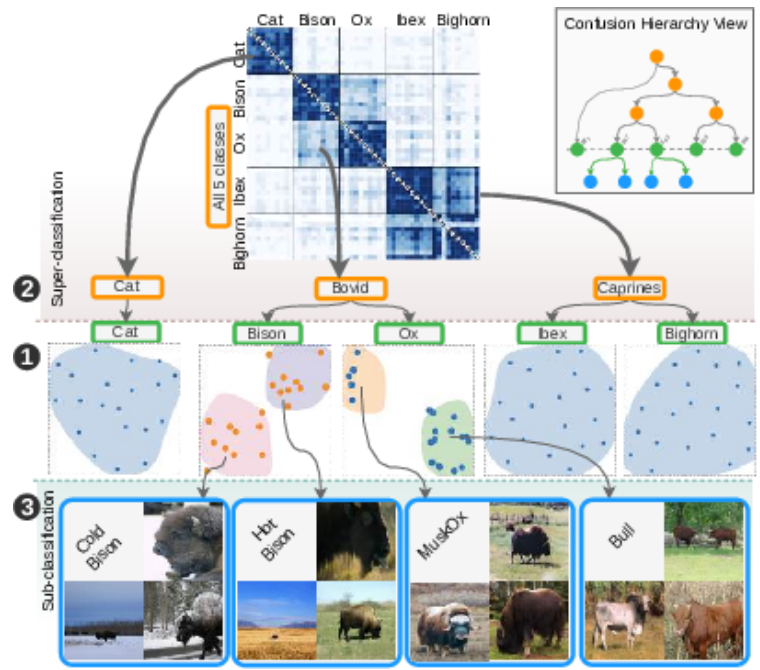

## Unimportant Channels Identification

Channels activating for unimportant features - attending to borders of the image.




## Confusion Hierarchical Generation with Reasoning

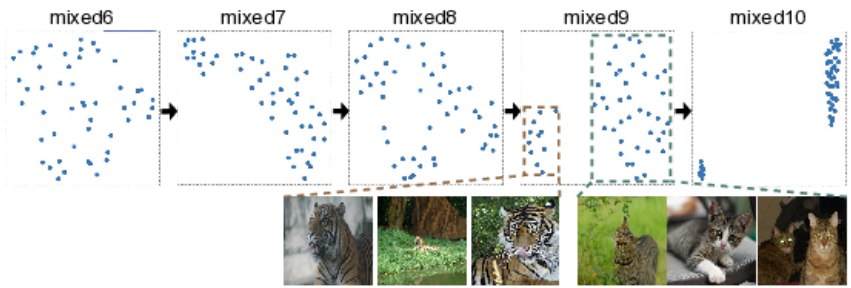
Starting from 5 classes, 3 super-classes (Cat, Bovid, and Caprine) are created. In the Bison's Scatterplot View, two clusters show examples of Bison in Cold & Hot weather. Similarly, Oxen's Scatterplot View shows furry and barrel-shaped bodies (MuskOx breed), and less furry cow-like bodies (Bull); creating 2 subclasses.



# Applications of ChannelExplorer

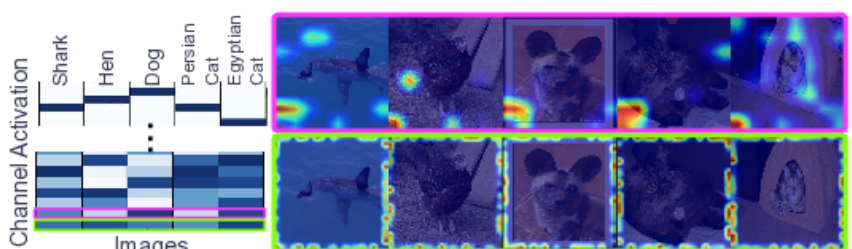
## Mislabel Identification

Scatterplot view of ChannelExplorer shows that a classification dataset have mislabeled inputs. In ImageNet, cats are labeled as Tiger Cat.



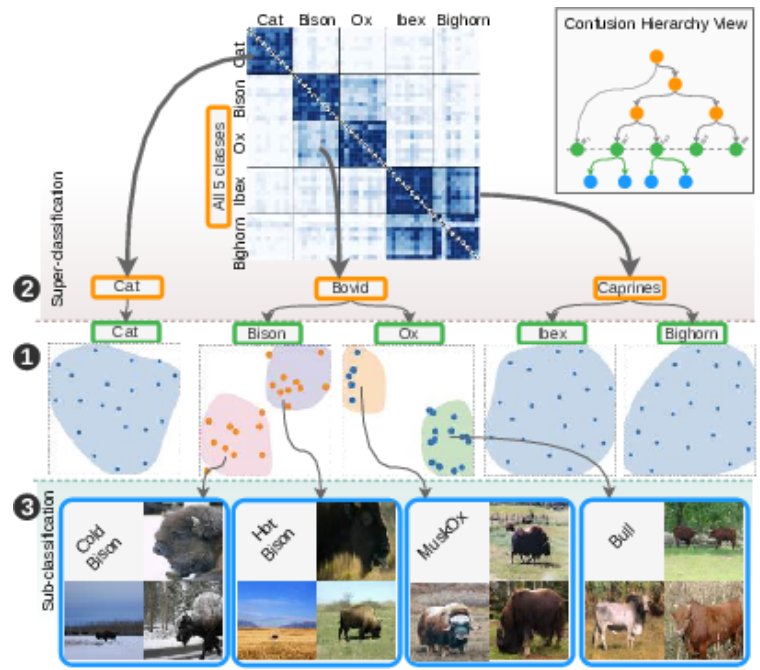

## Unimportant Channels Identification

Channels activating for unimportant features - attending to borders of the image.




## Confusion Hierarchical Generation with Reasoning

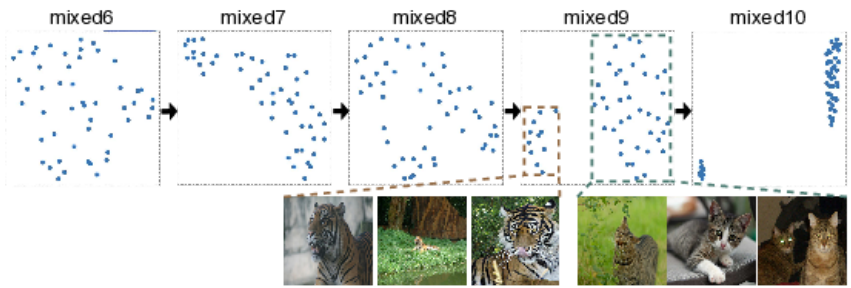
Starting from 5 classes, 3 super-classes (Cat, Bovid, and Caprine) are created. In the Bison's Scatterplot View, two clusters show examples of Bison in Cold & Hot weather. Similarly, Oxen's Scatterplot View shows furry and barrel-shaped bodies (MuskOx breed), and less furry cow-like bodies (Bull); creating 2 subclasses.



# Applications of ChannelExplorer

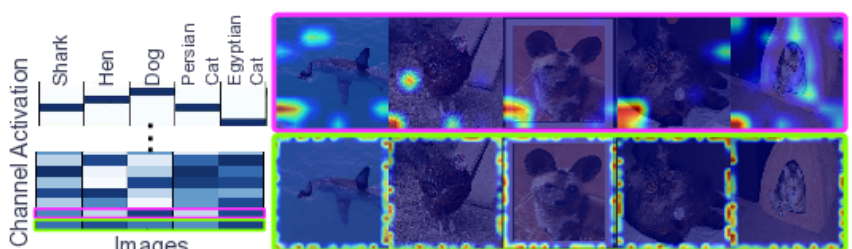
## Mislabel Identification

Scatterplot view of ChannelExplorer shows that a classification dataset have mislabeled inputs. In ImageNet, cats are labeled as Tiger Cat.



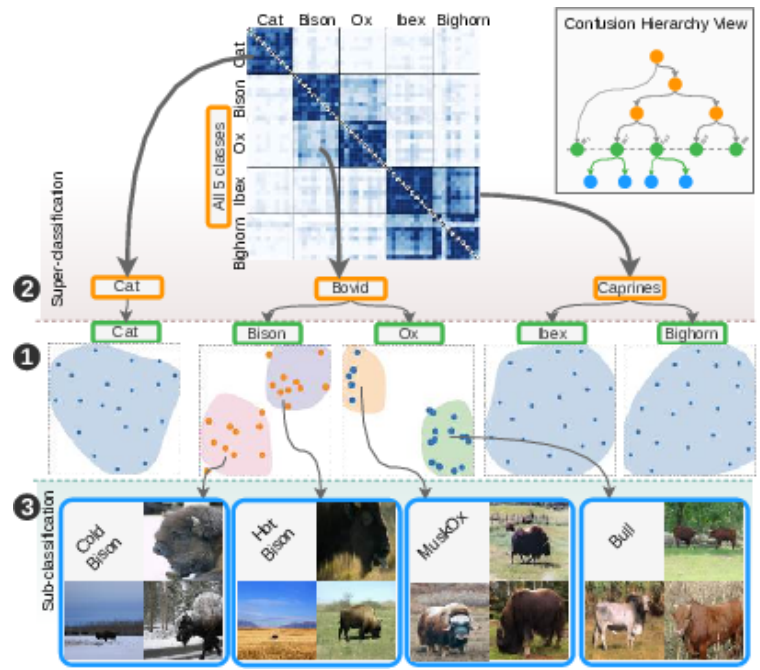

## Unimportant Channels Identification

Channels activating for unimportant features - attending to borders of the image.




## Confusion Hierarchical Generation with Reasoning

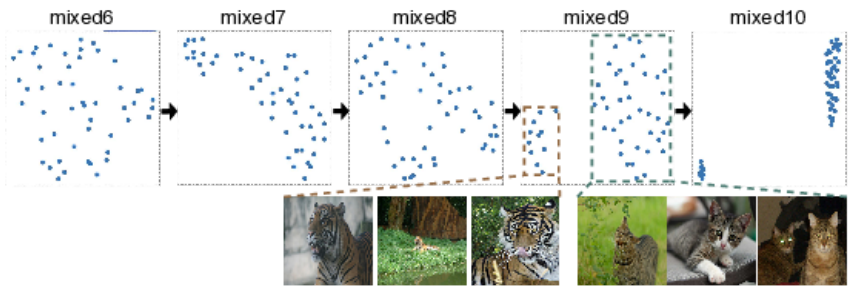
Starting from 5 classes, 3 super-classes (Cat, Bovid, and Caprine) are created. In the Bison's Scatterplot View, two clusters show examples of Bison in Cold & Hot weather. Similarly, Oxen's Scatterplot View shows furry and barrel-shaped bodies (MuskOx breed), and less furry cow-like bodies (Bull); creating 2 subclasses.



# Applications of ChannelExplorer

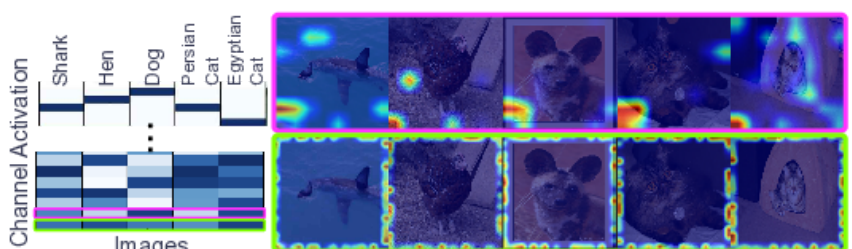
## Mislabel Identification

Scatterplot view of ChannelExplorer shows that a classification dataset have mislabeled inputs. In ImageNet, cats are labeled as Tiger Cat.



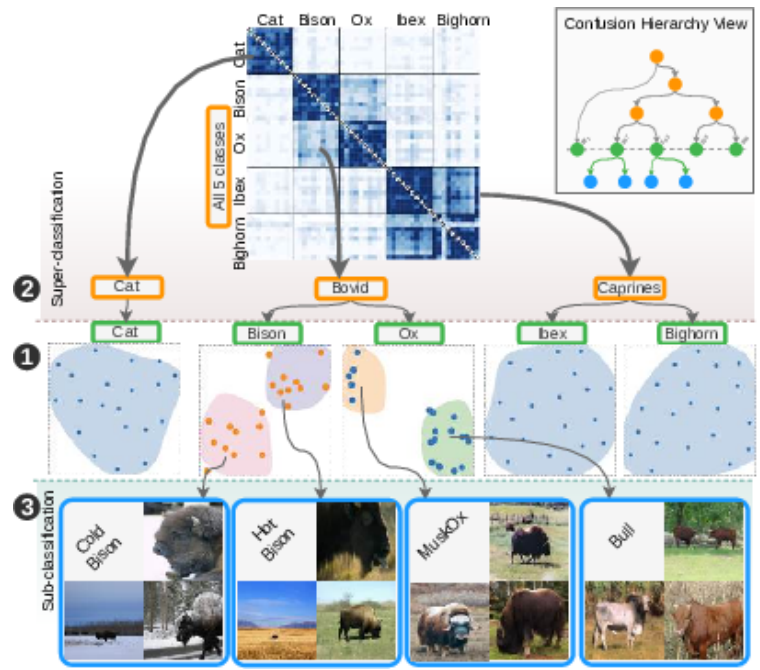

## Unimportant Channels Identification

Channels activating for unimportant features - attending to borders of the image.




## Confusion Hierarchical Generation with Reasoning

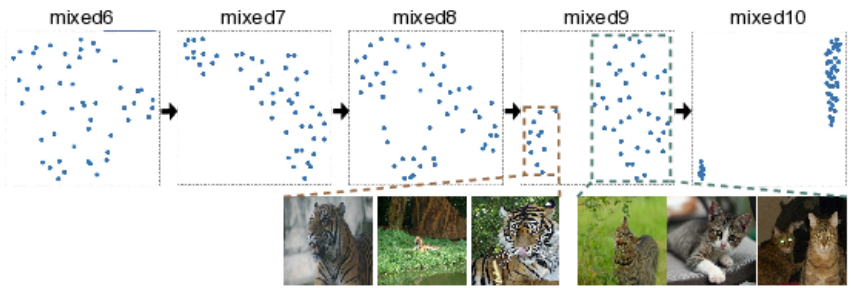
Starting from 5 classes, 3 super-classes (Cat, Bovid, and Caprine) are created. In the Bison's Scatterplot View, two clusters show examples of Bison in Cold & Hot weather. Similarly, Oxen's Scatterplot View shows furry and barrel-shaped bodies (MuskOx breed), and less furry cow-like bodies (Bull); creating 2 subclasses.



# Applications of ChannelExplorer

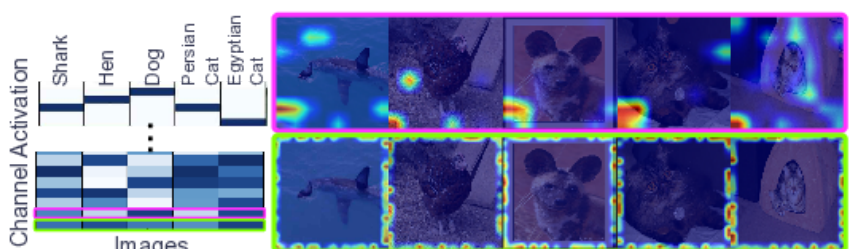
## Mislabel Identification

Scatterplot view of ChannelExplorer shows that a classification dataset have mislabeled inputs. In ImageNet, cats are labeled as Tiger Cat.



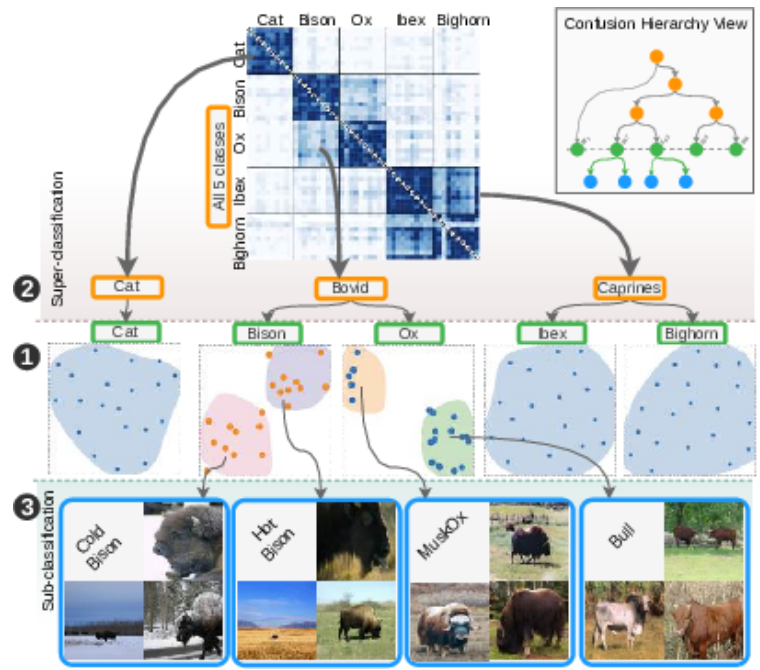

## Unimportant Channels Identification

Channels activating for unimportant features - attending to borders of the image.




## Confusion Hierarchical Generation with Reasoning

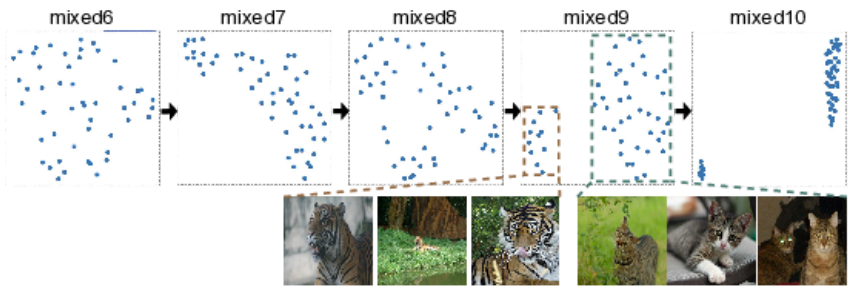
Starting from 5 classes, 3 super-classes (Cat, Bovid, and Caprine) are created. In the Bison's Scatterplot View, two clusters show examples of Bison in Cold & Hot weather. Similarly, Oxen's Scatterplot View shows furry and barrel-shaped bodies (MuskOx breed), and less furry cow-like bodies (Bull); creating 2 subclasses.



# Applications of ChannelExplorer

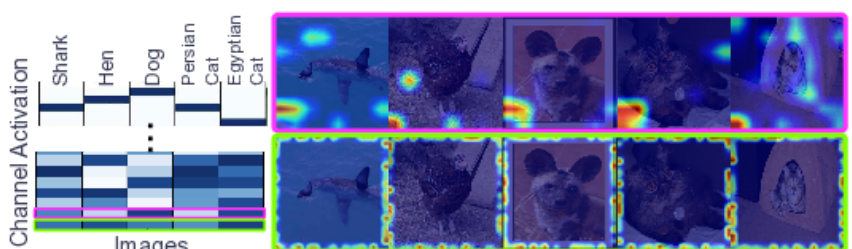
## Mislabel Identification

Scatterplot view of ChannelExplorer shows that a classification dataset have mislabeled inputs. In ImageNet, cats are labeled as Tiger Cat.



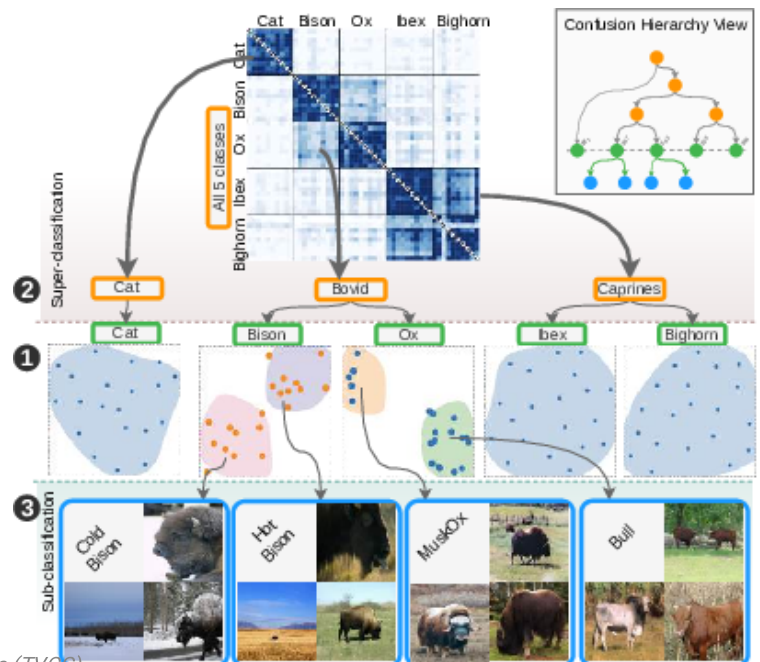

## Unimportant Channels Identification

Channels activating for unimportant features - attending to borders of the image.




## Confusion Hierarchical Generation with Reasoning

Starting from 5 classes, 3 super-classes (Cat, Bovid, and Caprine) are created. In the Bison's Scatterplot View, two clusters show examples of Bison in Cold & Hot weather. Similarly, Oxen's Scatterplot View shows furry and barrel-shaped bodies (MuskOx breed), and less furry cow-like bodies (Bull); creating 2 subclasses.

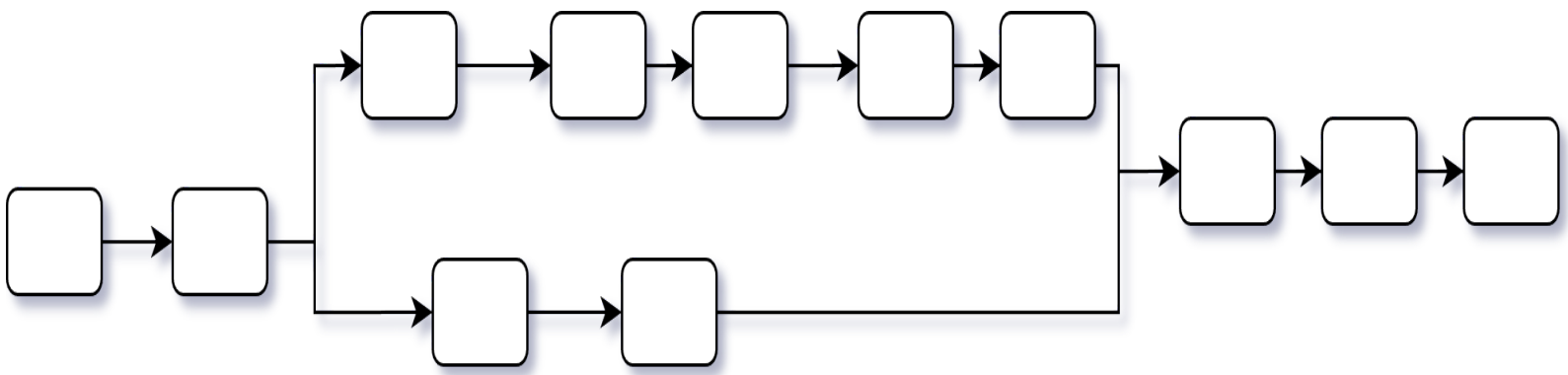


# Lessons of ChannelExplorer for Future Designs

- Performed **User Studies** with 10 participants.

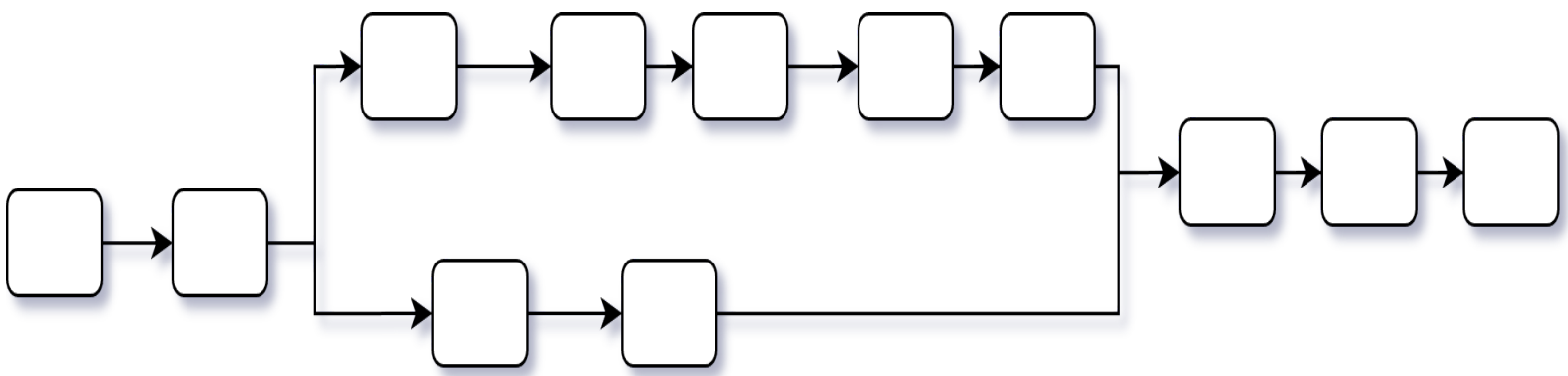
# Lessons of ChannelExplorer for Future Designs

- Performed **User Studies** with 10 participants.
- **Node-Link Diagram** is a powerful and flexible visualization method for any neural networks.



# Lessons of ChannelExplorer for Future Designs

- Performed **User Studies** with 10 participants.
- **Node-Link Diagram** is a powerful and flexible visualization method for any neural networks.
- Large models can clutter the visualization.



# Lessons of ChannelExplorer for Future Designs

- Performed **User Studies** with 10 participants.
- **Node-Link Diagram** is a powerful and flexible visualization m ne
- Large models can clutter the visualization.
- Group Layers to create **Hierarchical Graph Views**.

```
class ComplexNetWithBranch(torch.nn.Module):
    def __init__(self):
        super(ComplexNetWithBranch, self).__init__()

        # Common layers before branching
        self.seq = torch.nn.Sequential(
            torch.nn.Linear(10, 20),
            torch.nn.ReLU()
        )

        self.inner_seq_1 = torch.nn.Sequential(
            torch.nn.Linear(15, 15),
            torch.nn.ReLU()
        )
        self.inner_seq_2 = torch.nn.Sequential(
            torch.nn.Linear(15, 15),
            torch.nn.ReLU()
        )

        # Branch 1
        self.branch1 = torch.nn.Sequential(
            torch.nn.Linear(20, 15),
            self.inner_seq_1,
            self.inner_seq_2
        )

        # Branch 2
        self.branch2 = torch.nn.Sequential(
            torch.nn.Linear(20, 15),
            torch.nn.ReLU()
        )

        # Common layers after recombining
        self.fc_combine = torch.nn.Linear(30, 10)
        self.fc_output = torch.nn.Linear(10, 1)

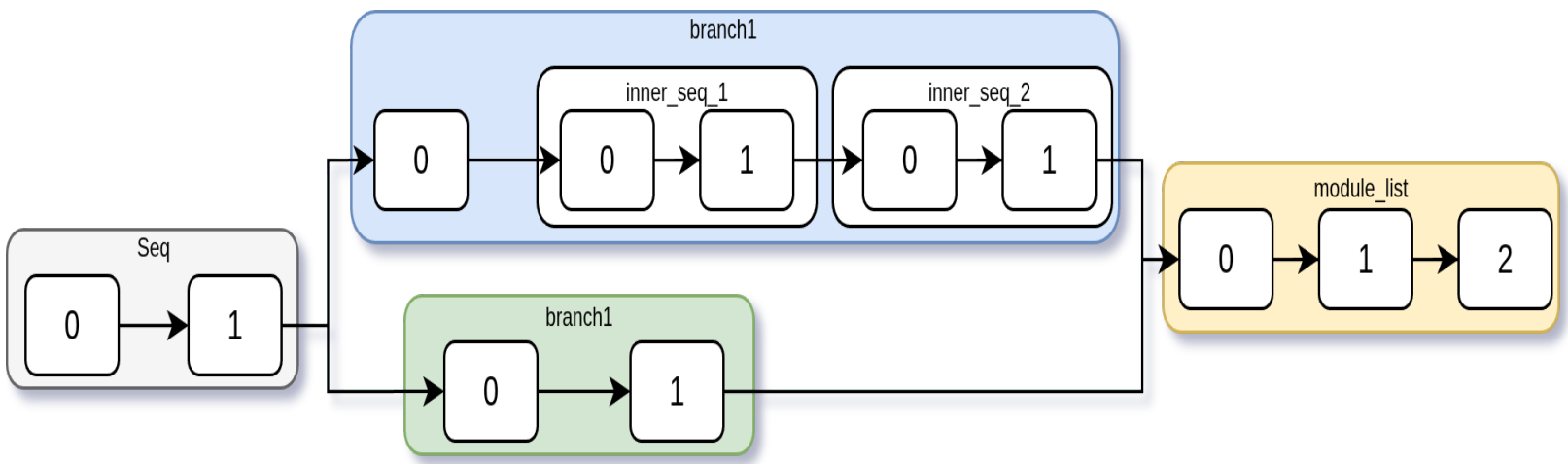
        # ModuleList after recombination
        self.module_list = torch.nn.ModuleList([
            torch.nn.Linear(1, 5),
            torch.nn.ReLU(),
            torch.nn.Linear(5, 1)
        ])

        self.branch_output = torch.nn.Linear(5, 5)
        self.branch_logit = torch.nn.ReLU()
```

PyTorch declaration

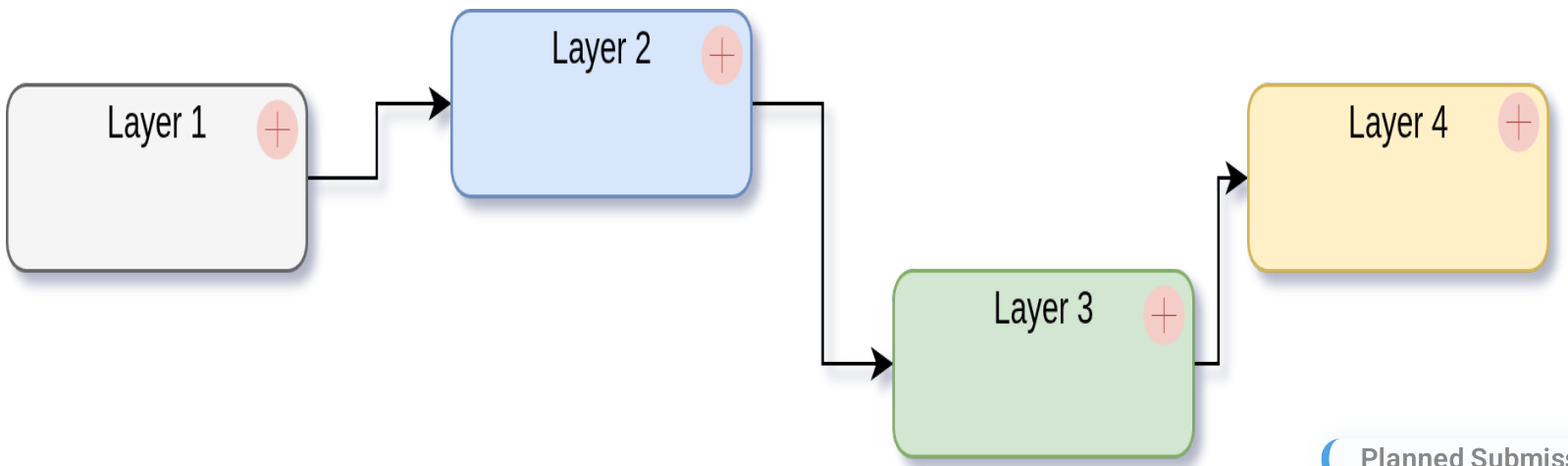
```
ComplexNetWithBranch(
  (seq): Sequential(
    (0): Linear(in_features=10, out_features=20, bias=True)
    (1): ReLU()
  )
  (inner_seq_1): Sequential(
    (0): Linear(in_features=15, out_features=15, bias=True)
    (1): ReLU()
  )
  (inner_seq_2): Sequential(
    (0): Linear(in_features=15, out_features=15, bias=True)
    (1): ReLU()
  )
  (branch1): Sequential(
    (0): Linear(in_features=20, out_features=15, bias=True)
    (1): Sequential(
      (0): Linear(in_features=15, out_features=15, bias=True)
      (1): ReLU()
    )
    (2): Sequential(
      (0): Linear(in_features=15, out_features=15, bias=True)
      (1): ReLU()
    )
  )
  (branch2): Sequential(
    (0): Linear(in_features=20, out_features=15, bias=True)
    (1): ReLU()
  )
  (fc_combine): Linear(in_features=30, out_features=10, bias=True)
  (fc_output): Linear(in_features=10, out_features=1, bias=True)
  (module_list): ModuleList(
    (0): Linear(in_features=1, out_features=5, bias=True)
    (1): ReLU()
    (2): Linear(in_features=5, out_features=1, bias=True)
  )
  (branch_output): Linear(in_features=5, out_features=5, bias=True)
  (branch_logit): ReLU()
)
```

PyTorch definition



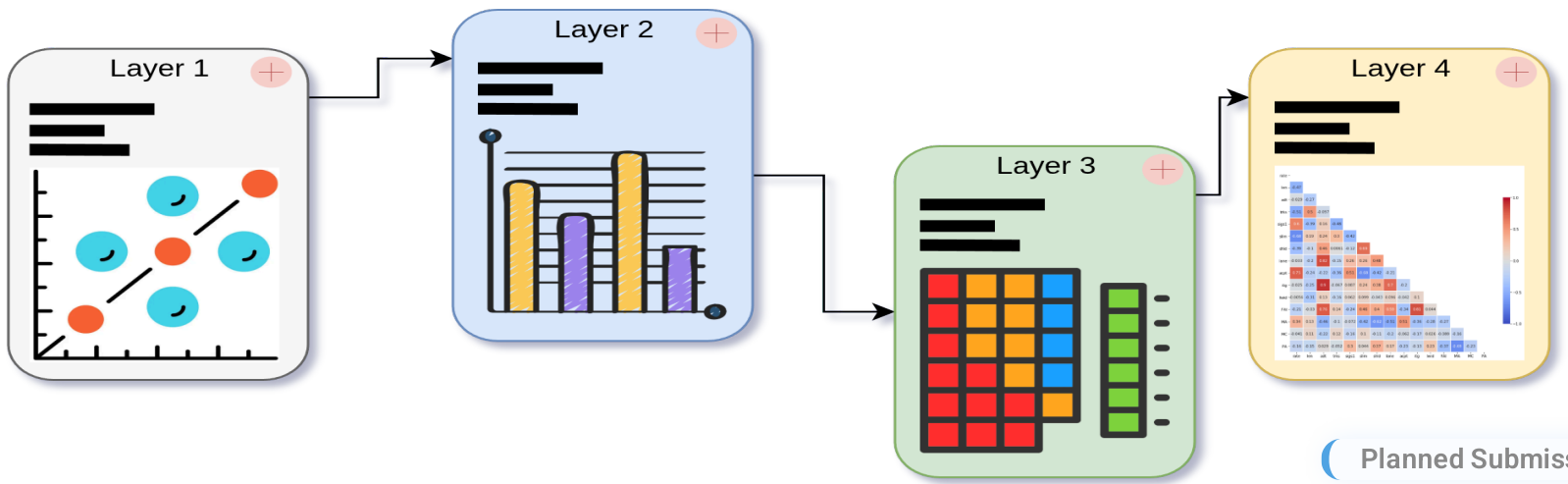
# Lessons of ChannelExplorer for Future Designs

- Performed **User Studies** with 10 participants.
- **Node-Link Diagram** is a powerful and flexible visualization method for any neural networks.
- Large models can clutter the visualization.
- Group Layers to create **Hierarchical Graph Views**.



# Lessons of ChannelExplorer for Future Designs

- Performed **User Studies** with 10 participants.
- **Node-Link Diagram** is a powerful and flexible visualization method for any neural networks.
- Large models can clutter the visualization.
- Group Layers to create **Hierarchical Graph Views**.
- **In Situ visualization** of tensor data per layer is scalably feasible.

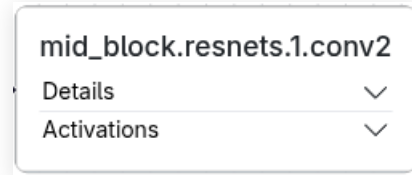


## Lessons of ChannelExplorer for Future Designs

- Performed **User Studies** with 10 participants.
- **Node-Link Diagram** is a powerful and flexible visualization method for any neural networks.
- Large models can clutter the visualization.
- Group Layers to create **Hierarchical Graph Views**.

**How can we use dynamic layer-based visualizations which can be created by users with low programming effort?**

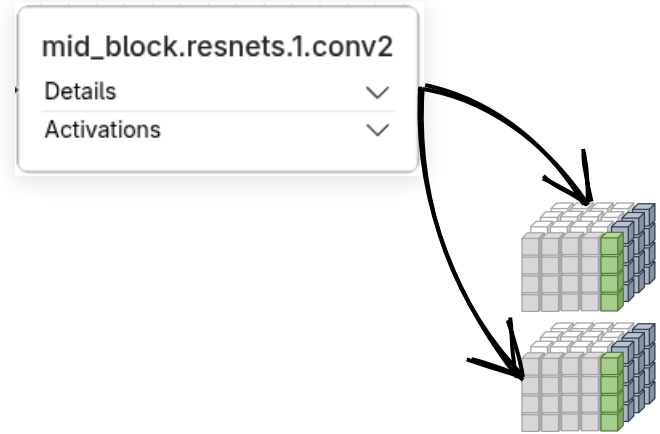
# Layer Visualization Grammar (LVG)




- A layer can **contain** other layers (expanded/collapsed).

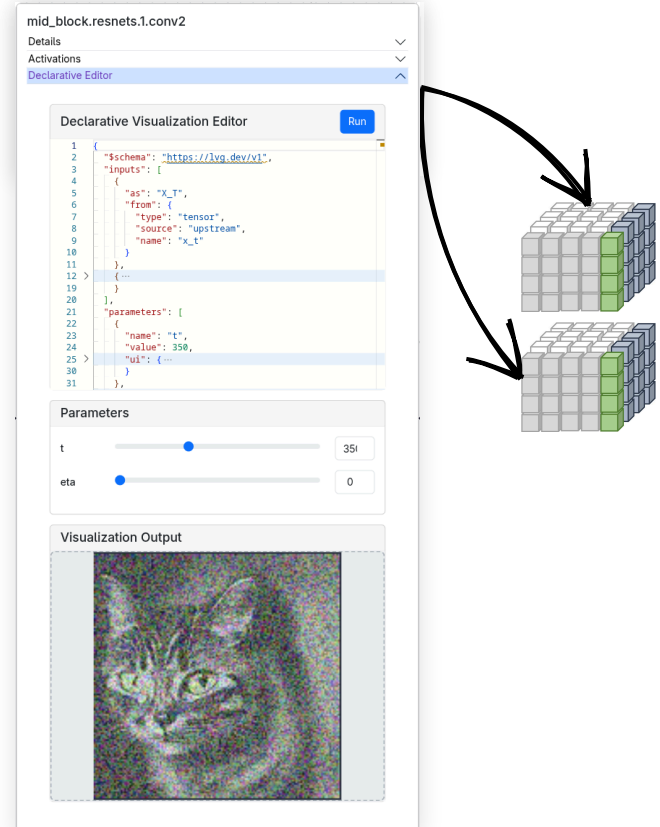
# Layer Visualization Grammar (LVG)

- A layer can **contain** other layers (expanded/collapsed).
- A leaf or collapsed layer can **emit** one or more tensors.



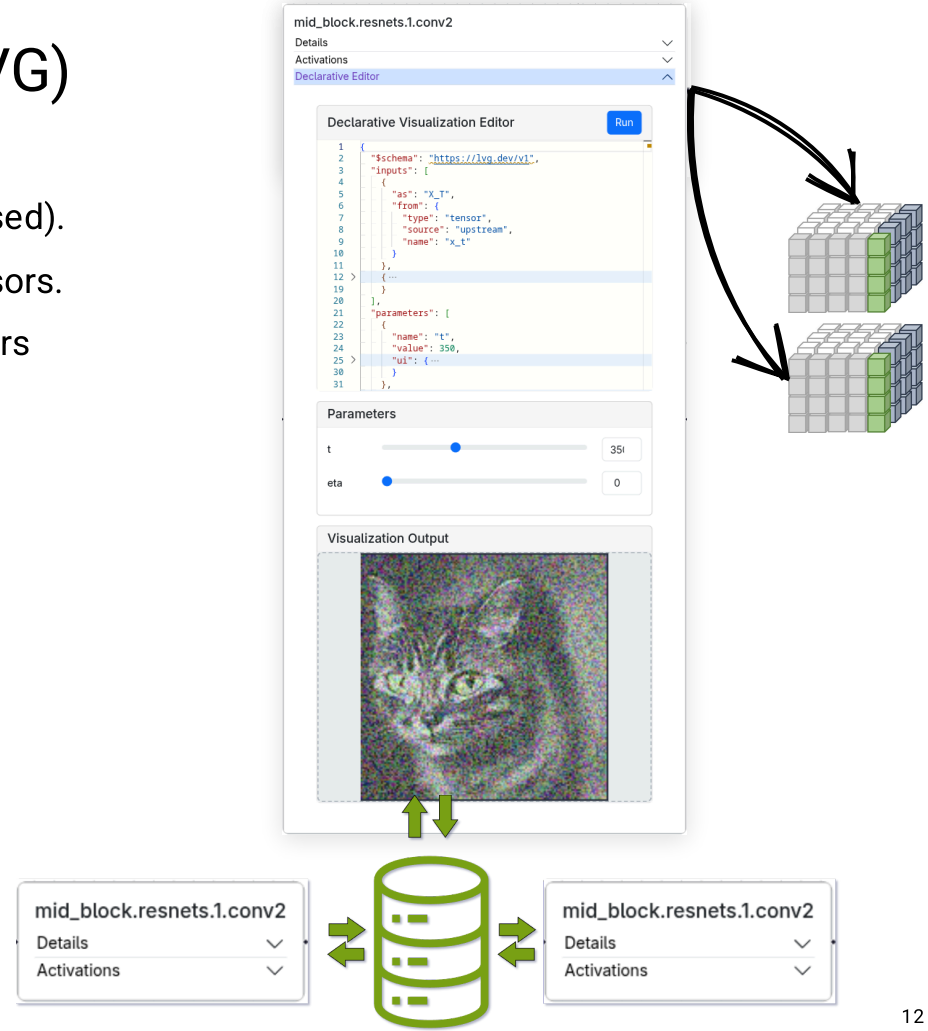
# Layer Visualization Grammar (LVG)

- A layer can **contain** other layers (expanded/collapsed).
- A leaf or collapsed layer can **emit** one or more tensors.
- Using a **declarative grammar**, we take these tensors
  - Apply **transformations** to the tensors
  - Apply **encodings**
  - Add **interactions**



# Layer Visualization Grammar (LVG)

- A layer can **contain** other layers (expanded/collapsed).
- A leaf or collapsed layer can **emit** one or more tensors.
- Using a **declarative grammar**, we take these tensors
  - Apply **transformations** to the tensors
  - Apply **encodings**
  - Add **interactions**
- LVG can put/pull data from global state.



* More details on LVG specification and types can be found in l.insane.casa/lvg.

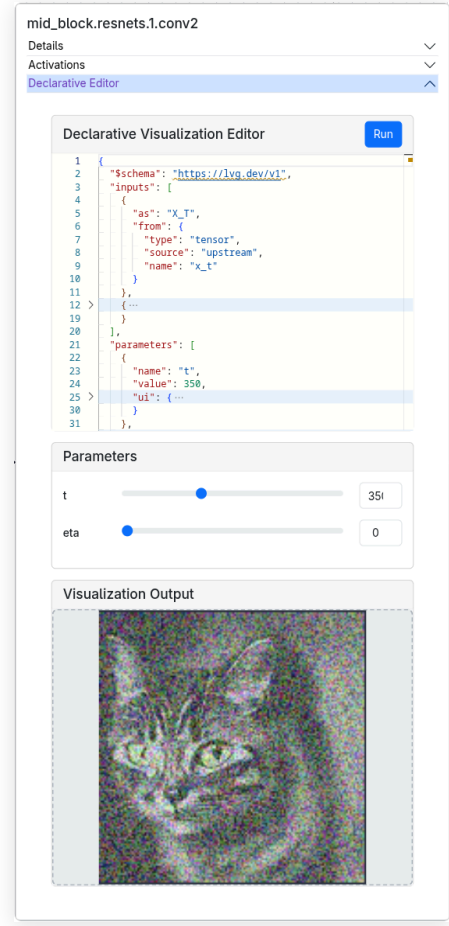
# Layer Visualization Grammar (Continued)

vega-lite.json

```
{
  "$schema": "https://vega.github.io
  "description": "A bubbleplot showi
  "data": [{
    "url": "data/cars.json",
    "transform": [
      {"type": "filter",
      "expr": "datum.year == year -
    ]
  }],
  "signals": [ ... ],
  "mark": "point",
  "encoding": {
    "x": {"field": "Horsepower", "ty
    "y": {"field": "Miles_per_Gallon
    "size": {"field": "Acceleration"
  }
}
```

layer.lvg.json

```
{
  "$schema": "https://lvg.dev/v1",
  "description": "Activated",
  "transform": [
    {"type": "expr",
    "expr": "curr_tensor\
              .summary().max() > 0.5",
    "name": "activated"}
  ]
  "signals": [ ... ],
  "mark": "rect",
  "encoding": {
    "x": {"field": "datum.i", ...},
    "y": {"value": 0}, "yOffset": 0,
    "color": ...,
  },
  "emit": [{
    "from": "activated",
    "as": "layer_i_activation_map"
  }],
}
```



* More details on LVG specification and types can be found in l.insane.casa/lvg.

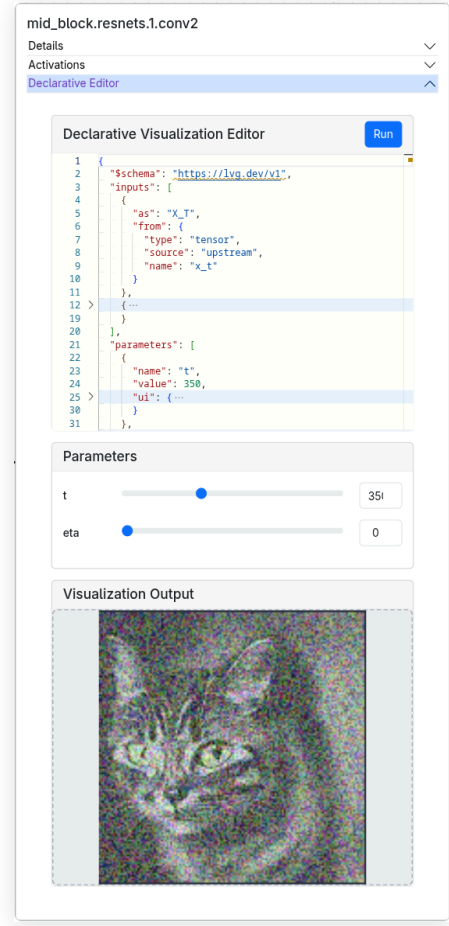
# Layer Visualization Grammar (Continued)

vega-lite.json

```
{
  "$schema": "https://vega.github.io
  "description": "A bubbleplot showi
  "data": [{
    "url": "data/cars.json",
    "transform": [
      {"type": "filter",
      "expr": "datum.year == year -
    ]
  }],
  "signals": [ ... ],
  "mark": "point",
  "encoding": {
    "x": {"field": "Horsepower", "ty
    "y": {"field": "Miles_per_Gallon
    "size": {"field": "Acceleration"
  }
}
```

layer.lvg.json

```
{
  "$schema": "https://lvg.dev/v1",
  "description": "Activated",
  "transform": [
    {"type": "expr",
    "expr": "curr_tensor\
              .summary().max() > 0.5",
    "name": "activated"}
  ]
  "signals": [ ... ],
  "mark": "rect",
  "encoding": {
    "x": {"field": "datum.i", ...},
    "y": {"value": 0}, "yOffset": 0,
    "color": ...,
  },
  "emit": [{
    "from": "activated",
    "as": "layer_i_activation_map"
  }],
}
```



* More details on LVG specification and types can be found in l.insane.casa/lvg.

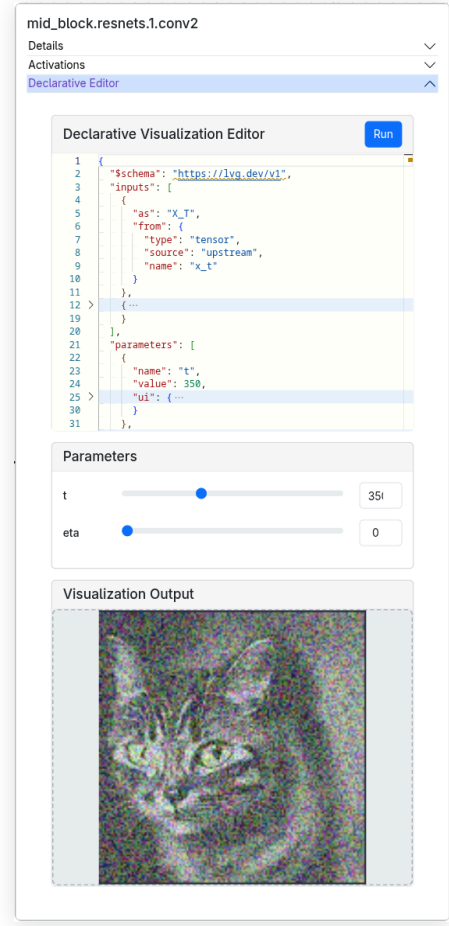
# Layer Visualization Grammar (Continued)

vega-lite.json

```
{
  "$schema": "https://vega.github.io
  "description": "A bubbleplot showi
  "data": [{
    "url": "data/cars.json",
    "transform": [
      {"type": "filter",
      "expr": "datum.year == year -
    ]
  }],
  "signals": [ ... ],
  "mark": "point",
  "encoding": {
    "x": {"field": "Horsepower", "ty
    "y": {"field": "Miles_per_Gallon
    "size": {"field": "Acceleration"
  }
}
```

layer.lvg.json

```
{
  "$schema": "https://lvg.dev/v1",
  "description": "t-SNE Projection",
  "transform": [
    {"type": "t-sne",
    "on": "curr_tensor",
    "batch_dim": 0, ...}
  ]
  "signals": [ ... ],
  "mark": "rect",
  "encoding": {
    "x": {"field": "datum.i", ...},
    "y": {"value": 0}, "yOffset": 0,
    "color": ...,
  },
  "emit": [{
    "from": "activated",
    "as": "layer_i_activation_map"
  }],
}
```



* More details on LVG specification and types can be found in l.insane.casa/lvg.

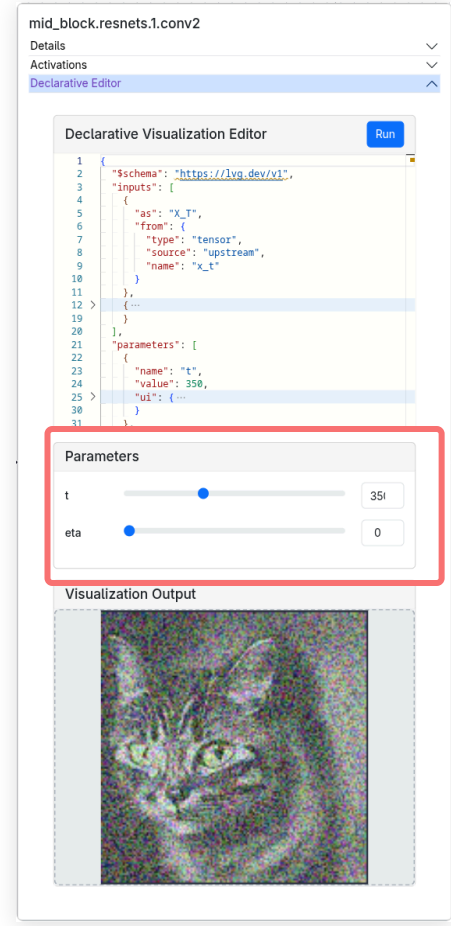
# Layer Visualization Grammar (Continued)

vega-lite.json

```
{
  "$schema": "https://vega.github.io
  "description": "A bubbleplot showi
  "data": [{
    "url": "data/cars.json",
    "transform": [
      {"type": "filter",
      "expr": "datum.year == year -
    ]
  }],
  "signals": [ ... ],
  "mark": "point",
  "encoding": {
    "x": {"field": "Horsepower", "ty
    "y": {"field": "Miles_per_Gallon
    "size": {"field": "Acceleration"
  }
}
```

layer.lvg.json

```
{
  "$schema": "https://lvg.dev/v1",
  "description": "t-SNE Projection",
  "transform": [
    {"type": "t-sne",
    "on": "curr_tensor",
    "batch_dim": 0, ...}
  ]
  "signals": [ ... ],
  "mark": "rect",
  "encoding": {
    "x": {"field": "datum.i", ...},
    "y": {"value": 0}, "yOffset": 0,
    "color": ...,
  },
  "emit": [{
    "from": "activated",
    "as": "layer_i_activation_map"
  }],
}
```



* More details on LVG specification and types can be found in l.insane.casa/lvg.

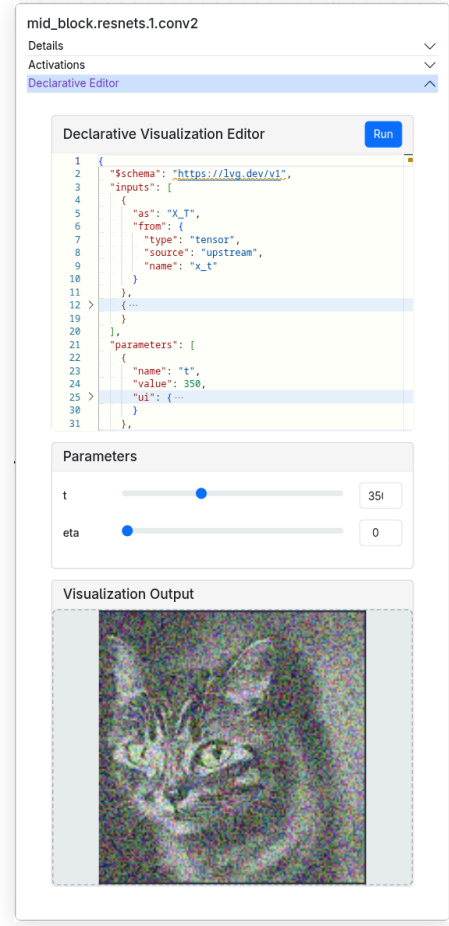
# Layer Visualization Grammar (Continued)

{} vega-lite.json

```
{
  "$schema": "https://vega.github.io
  "description": "A bubbleplot showi
  "data": [{
    "url": "data/cars.json",
    "transform": [
      {"type": "filter",
      "expr": "datum.year == year -
    ]
  }],
  "signals": [ ... ],
  "mark": "point",
  "encoding": {
    "x": {"field": "Horsepower", "ty
    "y": {"field": "Miles_per_Gallon
    "size": {"field": "Acceleration"
  }
}
```

{} layer.lvg.json

```
{
  "$schema": "https://lvg.dev/v1",
  "description": "t-SNE Projection",
  "transform": [
    {"type": "t-sne",
    "on": "curr_tensor",
    "batch_dim": 0, ...}
  ]
  "signals": [ ... ],
  "mark": "rect",
  "encoding": {
    "x": {"field": "datum.i", ...},
    "y": {"value": 0}, "yOffset": 0,
    "color": ...,
  },
  "emit": [{
    "from": "activated",
    "as": "layer_i_activation_map"
  }],
}
```



* More details on LVG specification and types can be found in l.insane.casa/lvg.

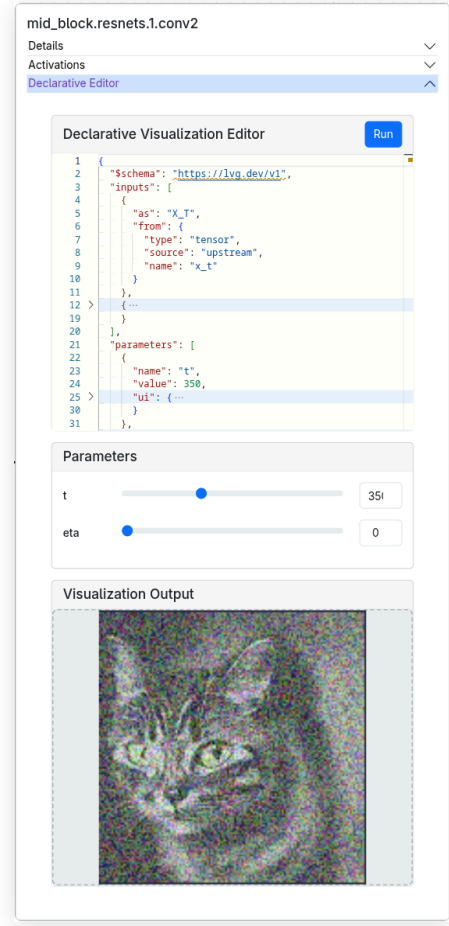
# Layer Visualization Grammar (Continued)

vega-lite.json

```
{
  "$schema": "https://vega.github.io
  "description": "A bubbleplot showi
  "data": [{
    "url": "data/cars.json",
    "transform": [
      {"type": "filter",
      "expr": "datum.year == year -
    ]
  }],
  "signals": [ ... ],
  "mark": "point",
  "encoding": {
    "x": {"field": "Horsepower", "ty
    "y": {"field": "Miles_per_Gallon
    "size": {"field": "Acceleration"
  }
}
```

layer.lvg.json

```
{
  "$schema": "https://lvg.dev/v1",
  "description": "t-SNE Projection",
  "transform": [
    {"type": "t-sne",
    "on": "curr_tensor",
    "batch_dim": 0, ...}
  ]
  "signals": [ ... ],
  "mark": "rect",
  "encoding": {
    "x": {"field": "datum.i", ...},
    "y": {"value": 0}, "yOffset": 0,
    "color": ...,
  },
  "emit": [{
    "from": "activated",
    "as": "layer_i_activation_map"
  }],
}
```



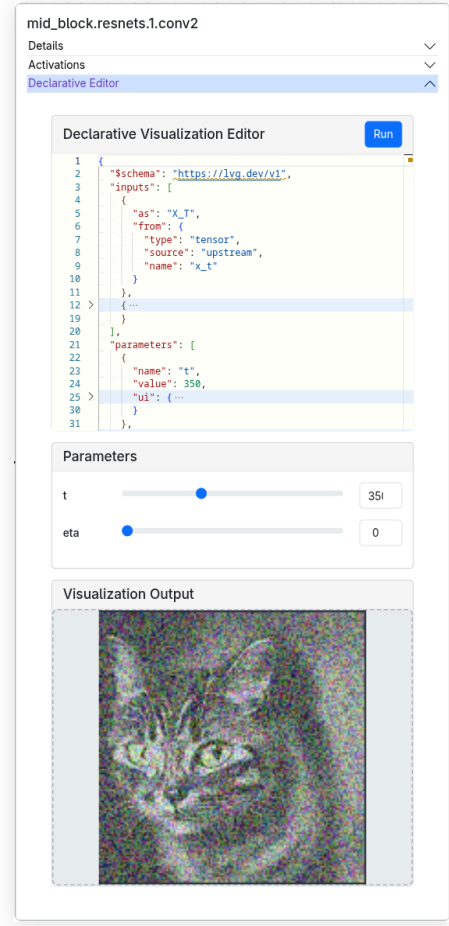
# Layer Visualization Grammar (Continued)

vega-lite.json

```
{
  "$schema": "https://vega.github.io
  "description": "A bubbleplot showi
  "data": [{
    "url": "data/cars.json",
    "transform": [
      {"type": "filter",
      "expr": "datum.year == year -
    ]
  }],
  "signals": [ ... ],
  "mark": "point",
  "encoding": {
    "x": {"field": "Horsepower", "ty
    "y": {"field": "Miles_per_Gallon
    "size": {"field": "Acceleration"
  }
}
```

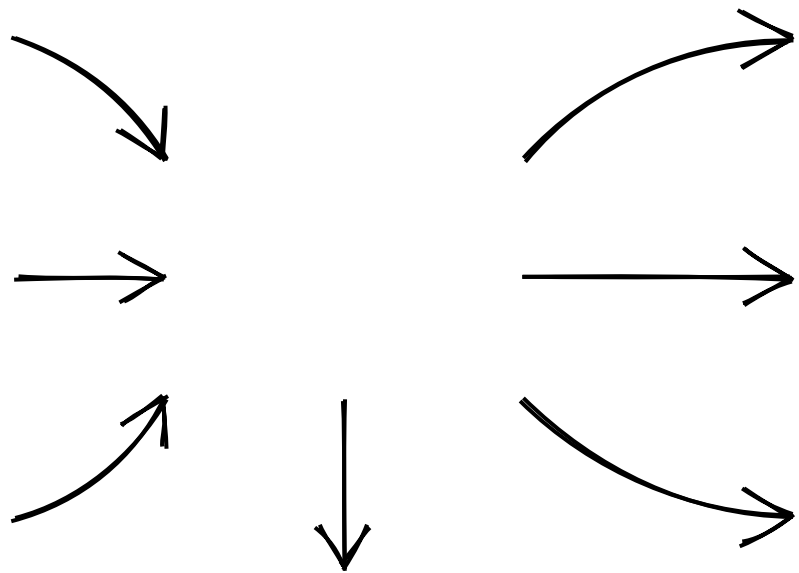
layer.lvg.json

```
{
  "$schema": "https://lvg.dev/v1",
  "description": "t-SNE Projection",
  "transform": [
    {"type": "t-sne",
    "on": "curr_tensor",
    "batch_dim": 0, ...}
  ]
  "signals": [ ... ],
  "mark": "rect",
  "encoding": {
    "x": {"field": "datum.i", ...},
    "y": {"value": 0}, "yOffset": 0,
    "color": ...,
  },
  "emit": [{
    "from": "activated",
    "as": "layer_i_activation_map"
  }],
}
```

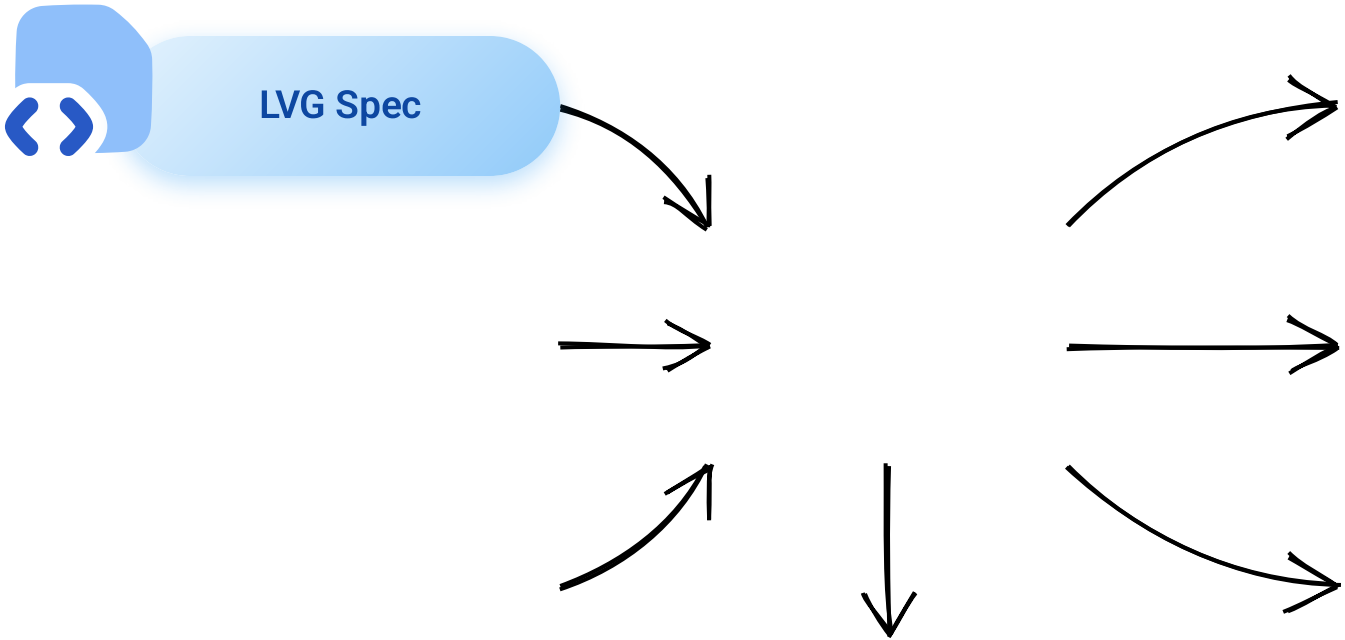



* Vega-Lite annotations grammar extension, IEEE VIS'25

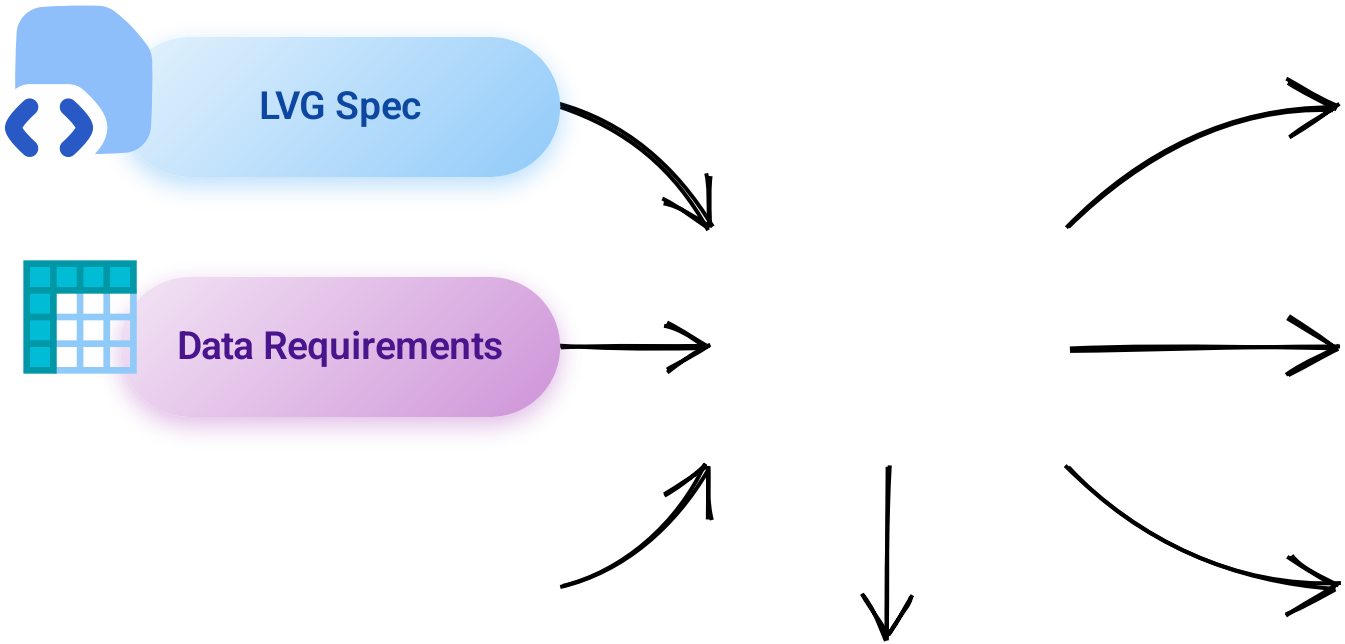
# Collaborative Share



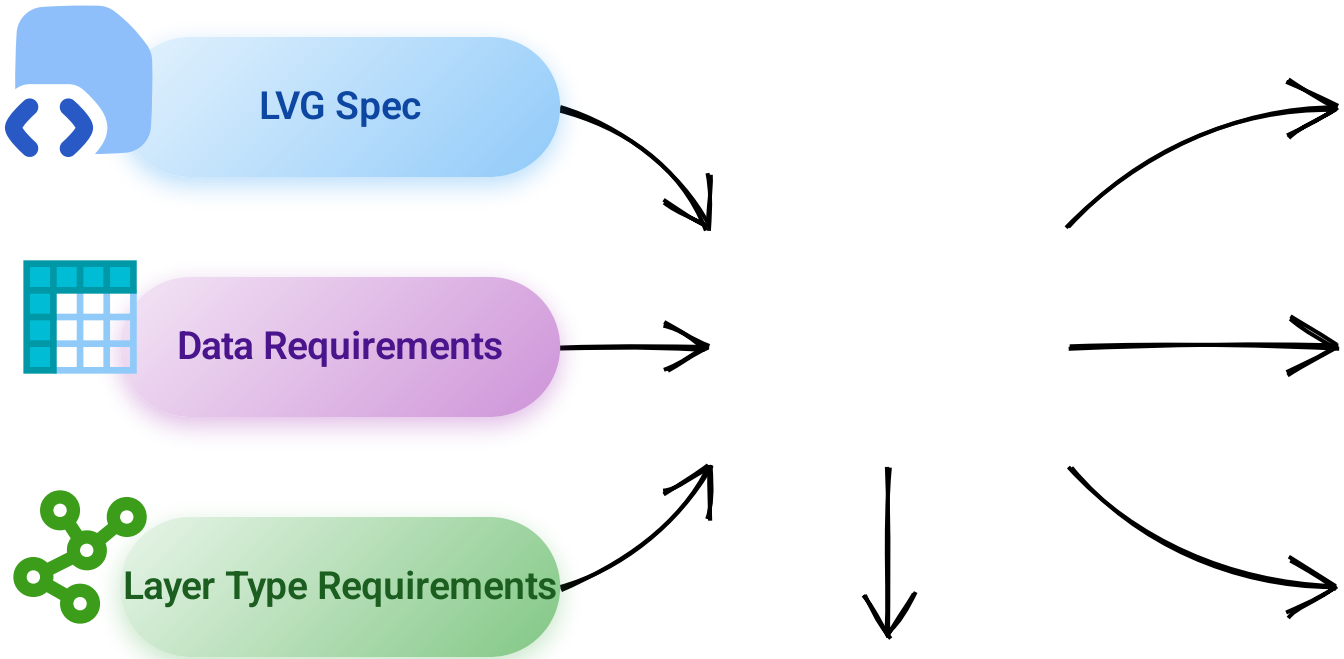
# Collaborative Share



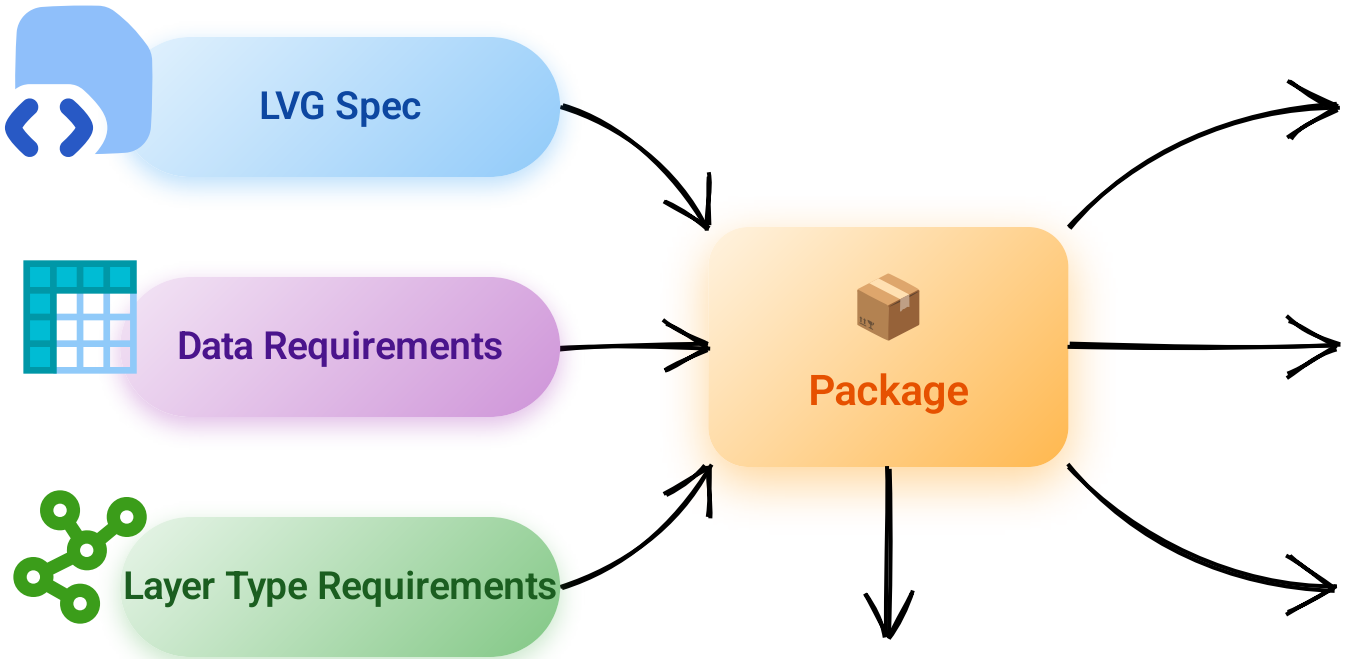
# Collaborative Share



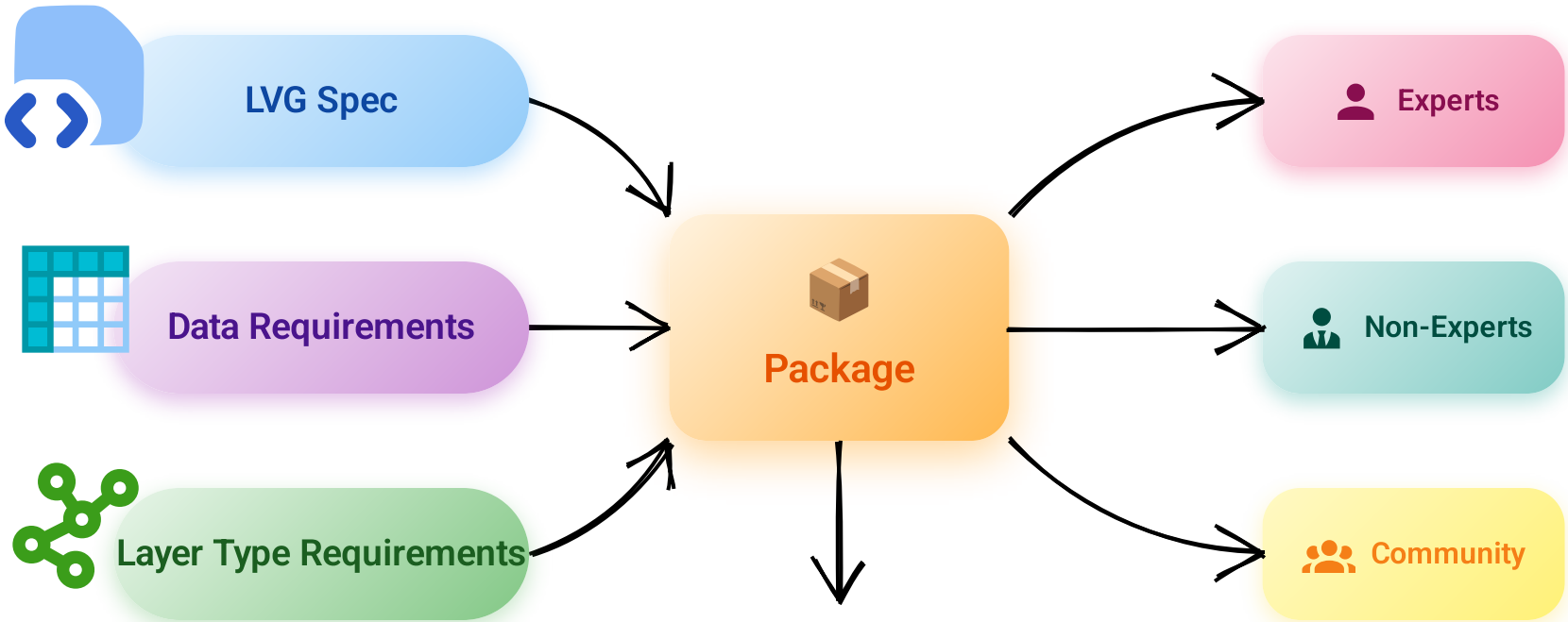
# Collaborative Share



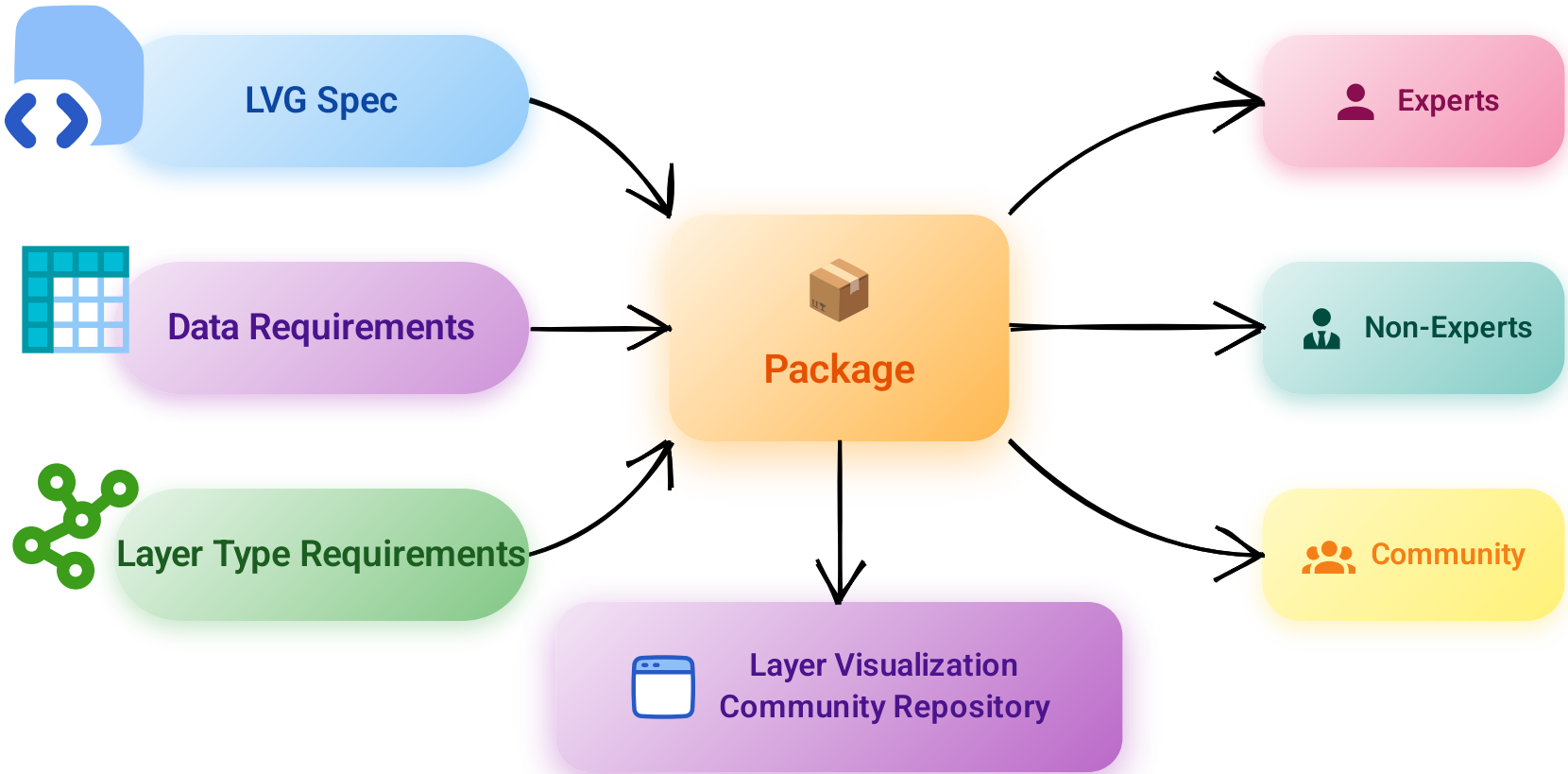
# Collaborative Share



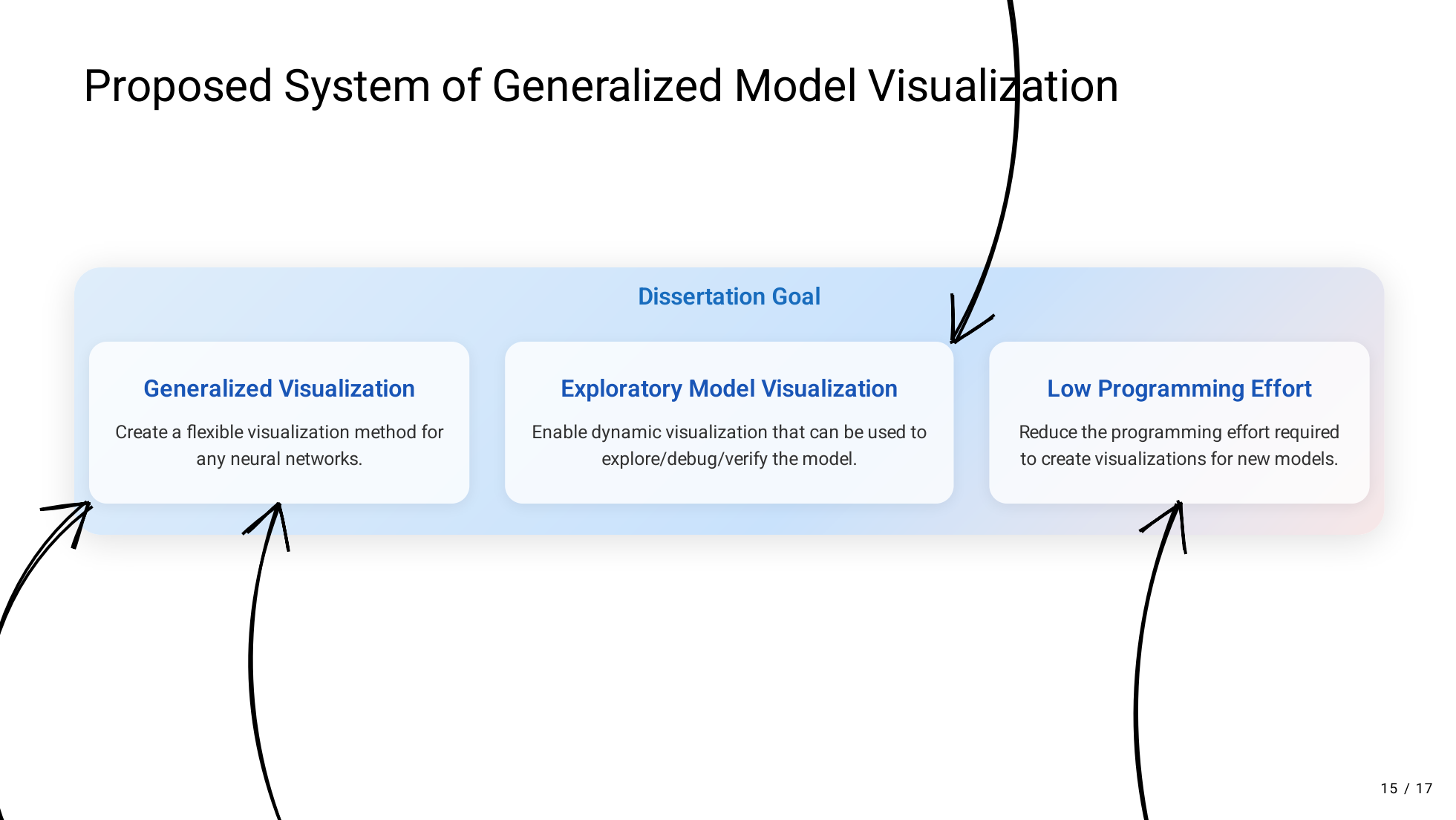
# Collaborative Share



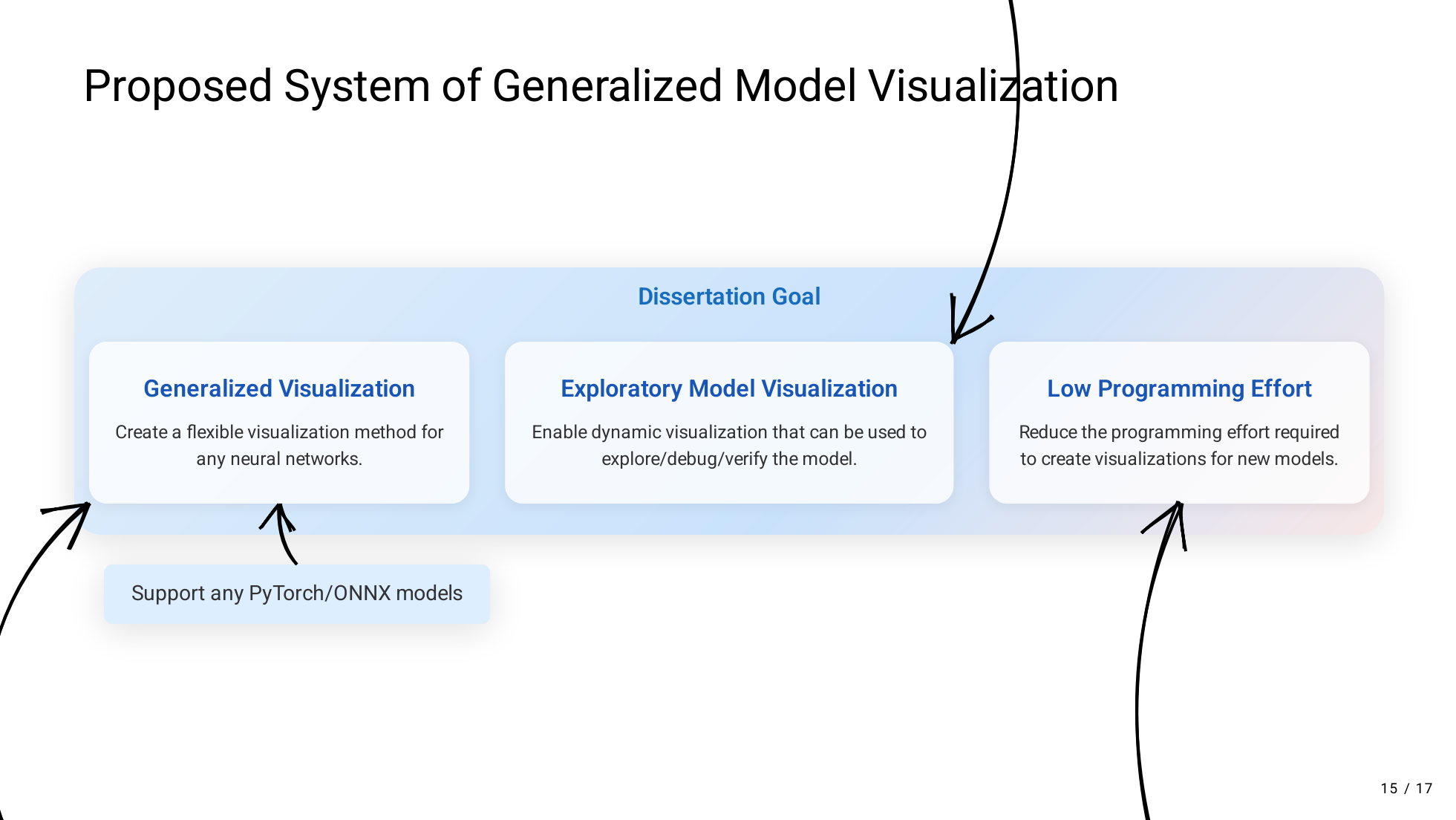
# Collaborative Share



# Proposed System of Generalized Model Visualization



# Proposed System of Generalized Model Visualization



# Proposed System of Generalized Model Visualization

## Dissertation Goal

### Generalized Visualization

Create a flexible visualization method for any neural networks.

### Exploratory Model Visualization

Enable dynamic visualization that can be used to explore/debug/verify the model.

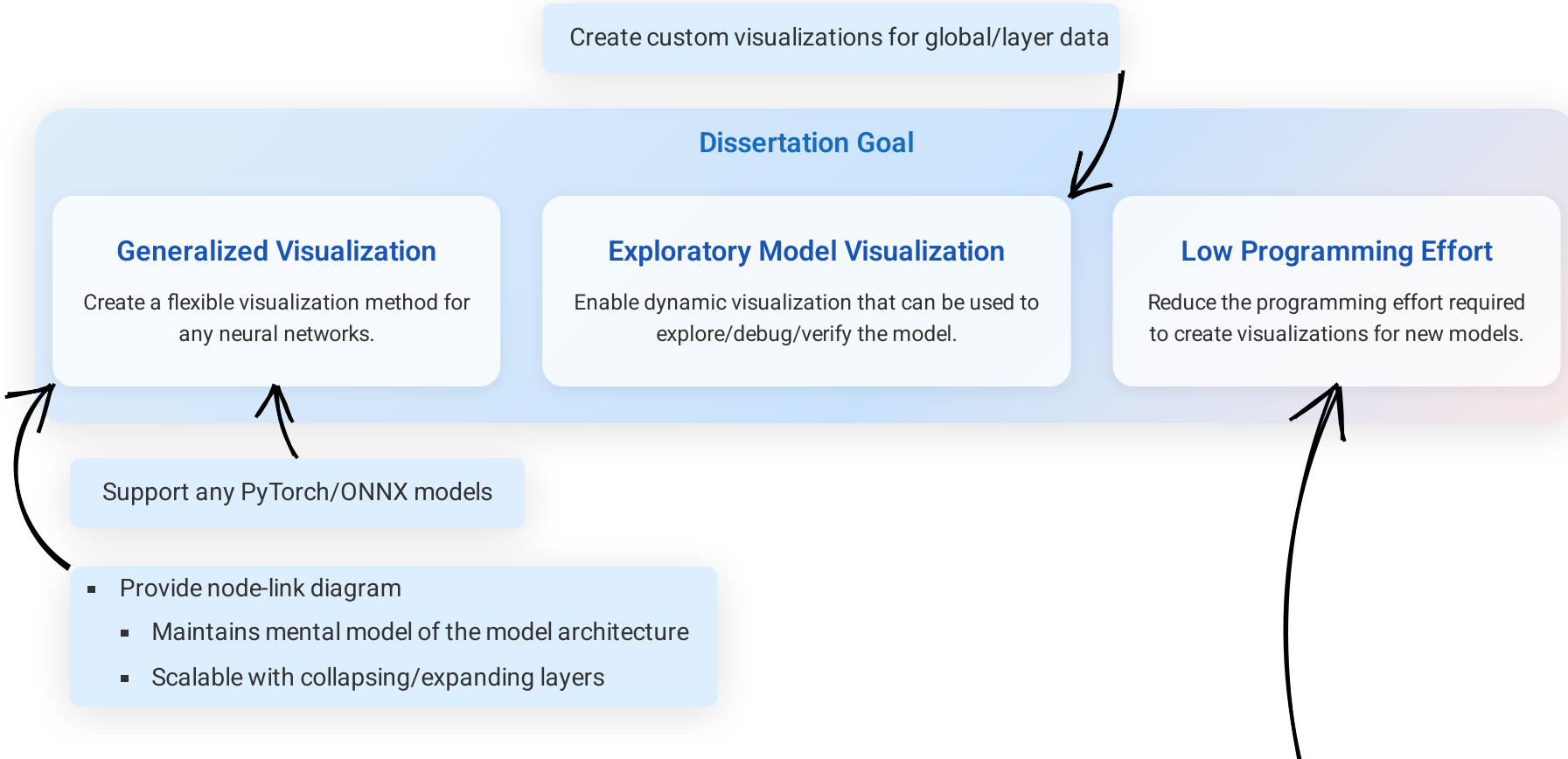
### Low Programming Effort

Reduce the programming effort required to create visualizations for new models.

Support any PyTorch/ONNX models

- Provide node-link diagram
  - Maintains mental model of the model architecture
  - Scalable with collapsing/expanding layers

# Proposed System of Generalized Model Visualization



# Proposed System of Generalized Model Visualization

Create custom visualizations for global/layer data

**Dissertation Goal**

### Generalized Visualization

Create a flexible visualization method for any neural networks.

### Exploratory Model Visualization

Enable dynamic visualization that can be used to explore/debug/verify the model.

### Low Programming Effort

Reduce the programming effort required to create visualizations for new models.

Support any PyTorch/ONNX models

- Provide node-link diagram
  - Maintains mental model of the model architecture
  - Scalable with collapsing/expanding layers

Design Declarative Grammar for layer visualizations

# Questions for DC Panelists

# Questions for DC Panelists

Q1. Is there any element lacking in the current proposal? Are there further studies or methodologies to enhance its applicability?

# Questions for DC Panelists

Q1. Is there any element lacking in the current proposal? Are there further studies or methodologies to enhance its applicability?

Q2. Does Layer Visualization Grammar with helper functions actually reduce programming knowledge required? Are there other shareable frameworks for generalized visualization?

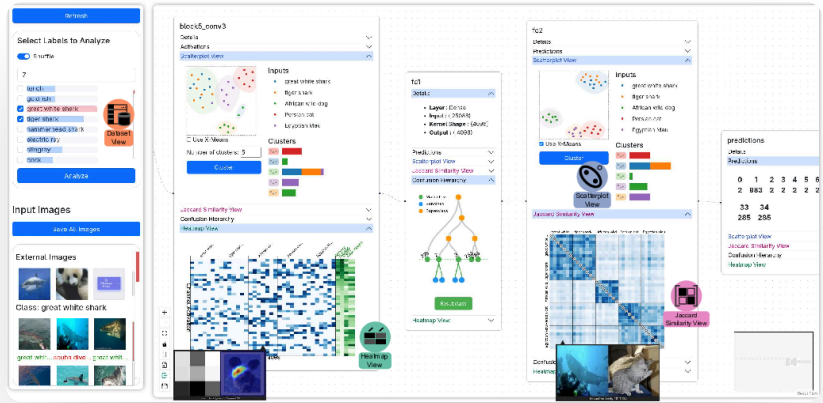
# Questions for DC Panelists

Q1. Is there any element lacking in the current proposal? Are there further studies or methodologies to enhance its applicability?

Q2. Does Layer Visualization Grammar with helper functions actually reduce programming knowledge required? Are there other shareable frameworks for generalized visualization?

Q3. Are my dissertation goals sufficiently defined, and can they be achieved through the proposed system design and implementation?

# Thank You!




## Towards a Generalized Deep Learning Visual Analytics Design


**Rahat Zaman**

rahat.zaman@utah.edu

rahatzaman.me


## Questions

Q1. How can I ensure that this design generalizes across a broader range of deep learning tasks and domains?

Q2. What are the key evaluation criteria to demonstrate the value and utility of my proposed system?

Q3. Are my dissertation goals sufficiently defined, and can they be achieved through the proposed system design and implementation?