Traffic Sign Recognition

In [2]:
```
# imports
import torch
import numpy as np
import matplotlib.pyplot as plt

from datetime import datetime
from model import Model
from torch.utils.tensorboard import SummaryWriter
from torch.utils.data import DataLoader

from utils.dataset import MappilaryDataset
from utils.tools import calculate_accuracy
from utils.vizualization import show_predictions, show_weights_for_class, plot_example_images
```

0. Setup dataset

In [3]:
```
# pseudo randomness
np.random.seed(0)
torch.manual_seed(0)

# datasets
batch_size = 100
train_dataset = MappilaryDataset('data/train.pkl')
test_dataset = MappilaryDataset('data/test.pkl')
val_dataset = MappilaryDataset('data/val.pkl')

# dataloaders
train_dataloader = DataLoader(train_dataset, batch_size=batch_size)
test_dataloader = DataLoader(test_dataset, batch_size=batch_size)
val_dataloader = DataLoader(val_dataset, batch_size=batch_size)
```

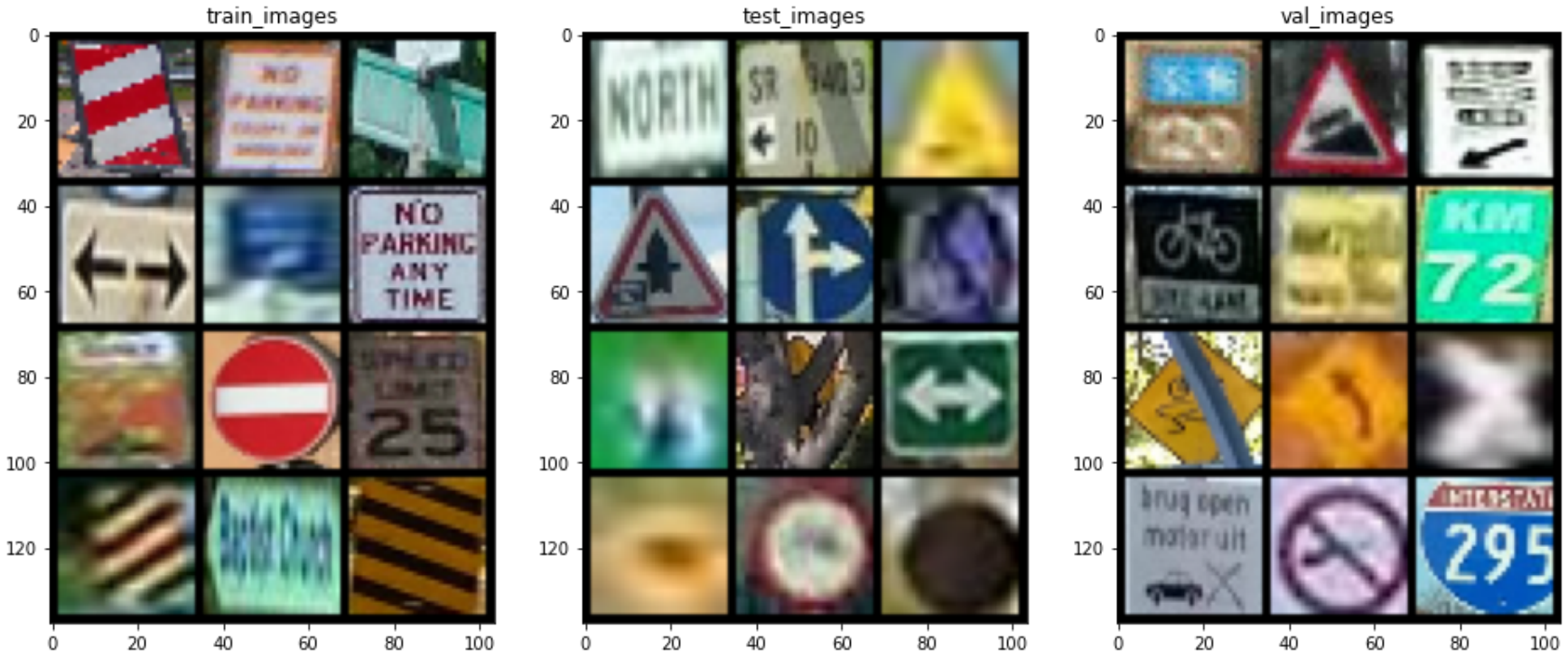
1. Plot some training examples

In [4]:
```
# 1. log some images
plt.figure(figsize=(16, 8))
```

```
plt.subplot(131)
plot_example_images('train_images', train_dataset)

plt.subplot(132)
plot_example_images('test_images', test_dataset)

plt.subplot(133)
plot_example_images('val_images', val_dataset)
```



In [5]:

```
# 2. Log some example labels
print(train_dataset.labels[:5])
print(test_dataset.labels[:5])
print(val_dataset.labels[:5])
```

```
['other-sign', 'regulatory--no-parking-or-no-stopping--g1', 'other-sign', 'complementary--extent-of-prohibition-area-both
-direction--g1', 'other-sign']
['other-sign', 'other-sign', 'other-sign', 'warning--crossroads--g1', 'regulatory--go-straight-or-turn-right--g1']
['other-sign', 'other-sign', 'other-sign', 'regulatory--bicycles-only--g3', 'other-sign']
```

2. Create the Model

In [6]:

```
# create model
model = Model()

# define loss & optimizer
loss_fn = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.01, momentum=0.9)

print(model)
```

```
Model(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=400, out_features=1024, bias=True)
  (fc2): Linear(in_features=1024, out_features=512, bias=True)
  (fc3): Linear(in_features=512, out_features=401, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
)
```

3. Train the Model

In [7]:

```
num_epoch = 10
train_tracker = []
test_tracker = []
val_tracker = []
for epoch in range(1, num_epoch + 1):
    model.train()
    running_train_acc = 0.0
    for i, (inputs, labels) in enumerate(train_dataloader):
        optimizer.zero_grad()  # Zero your gradients for every batch!
        outputs = model(inputs)  # Make predictions for this batch
        loss = loss_fn(outputs, labels)  # Compute the loss and its gradients
        loss.backward()
        optimizer.step()

        # calculate train accuracy
        _, y_pred = torch.max(outputs.data, 1)
        batch_train_accuracy = (y_pred == labels).sum().item()
        running_train_acc += batch_train_accuracy

        if i % 100 == 0 and i > 0:
            print(f"Epoch: {epoch} - Minibatch #{i} Accuracy: {batch_train_accuracy / batch_size:.2%}")
```

```
    train_accuracy = running_train_acc / len(train_dataset)

    # calculate test/val accuracy
    model.eval()
    test_accuracy = calculate_accuracy(model, test_dataloader)
    val_accuracy = calculate_accuracy(model, val_dataloader)

    train_tracker.append(train_accuracy)
    test_tracker.append(test_accuracy)
    val_tracker.append(val_accuracy)

    print(f"Finished processing epoch: {epoch}")
    print(f"Train Accuracy: {train_accuracy:.2%}")
    print(f"Test Accuracy: {test_accuracy:.2%}")
    print(f"Val Accuracy: {val_accuracy:.2%}")

print("Done training, saving the model..")
model_scripted = torch.jit.script(model)       # Export to TorchScript
model_scripted.save('models/latest_model.pt')  # Save to disk
```

```
Epoch: 1 - Minibatch #100 Accuracy: 69.00%
Epoch: 1 - Minibatch #200 Accuracy: 75.00%
Epoch: 1 - Minibatch #300 Accuracy: 69.00%
Epoch: 1 - Minibatch #400 Accuracy: 63.00%
Epoch: 1 - Minibatch #500 Accuracy: 60.00%
Epoch: 1 - Minibatch #600 Accuracy: 70.00%
Epoch: 1 - Minibatch #700 Accuracy: 66.00%
Epoch: 1 - Minibatch #800 Accuracy: 68.00%
Epoch: 1 - Minibatch #900 Accuracy: 66.00%
Epoch: 1 - Minibatch #1000 Accuracy: 61.00%
Epoch: 1 - Minibatch #1100 Accuracy: 65.00%
Epoch: 1 - Minibatch #1200 Accuracy: 77.00%
Epoch: 1 - Minibatch #1300 Accuracy: 63.00%
Epoch: 1 - Minibatch #1400 Accuracy: 68.00%
Finished processing epoch: 1
Train Accuracy: 66.51%
Test Accuracy: 69.73%
Val Accuracy: 69.42%
Epoch: 2 - Minibatch #100 Accuracy: 75.00%
Epoch: 2 - Minibatch #200 Accuracy: 79.00%
Epoch: 2 - Minibatch #300 Accuracy: 80.00%
Epoch: 2 - Minibatch #400 Accuracy: 73.00%
Epoch: 2 - Minibatch #500 Accuracy: 74.00%
Epoch: 2 - Minibatch #600 Accuracy: 78.00%
```

```
Epoch: 2 - Minibatch #700 Accuracy: 79.00%
Epoch: 2 - Minibatch #800 Accuracy: 78.00%
Epoch: 2 - Minibatch #900 Accuracy: 71.00%
Epoch: 2 - Minibatch #1000 Accuracy: 74.00%
Epoch: 2 - Minibatch #1100 Accuracy: 79.00%
Epoch: 2 - Minibatch #1200 Accuracy: 76.00%
Epoch: 2 - Minibatch #1300 Accuracy: 73.00%
Epoch: 2 - Minibatch #1400 Accuracy: 80.00%
Finished processing epoch: 2
Train Accuracy: 75.31%
Test Accuracy: 79.48%
Val Accuracy: 78.95%
Epoch: 3 - Minibatch #100 Accuracy: 81.00%
Epoch: 3 - Minibatch #200 Accuracy: 83.00%
Epoch: 3 - Minibatch #300 Accuracy: 83.00%
Epoch: 3 - Minibatch #400 Accuracy: 75.00%
Epoch: 3 - Minibatch #500 Accuracy: 77.00%
Epoch: 3 - Minibatch #600 Accuracy: 82.00%
Epoch: 3 - Minibatch #700 Accuracy: 90.00%
Epoch: 3 - Minibatch #800 Accuracy: 82.00%
Epoch: 3 - Minibatch #900 Accuracy: 79.00%
Epoch: 3 - Minibatch #1000 Accuracy: 77.00%
Epoch: 3 - Minibatch #1100 Accuracy: 82.00%
Epoch: 3 - Minibatch #1200 Accuracy: 82.00%
Epoch: 3 - Minibatch #1300 Accuracy: 80.00%
Epoch: 3 - Minibatch #1400 Accuracy: 82.00%
Finished processing epoch: 3
Train Accuracy: 80.78%
Test Accuracy: 82.56%
Val Accuracy: 82.15%
Epoch: 4 - Minibatch #100 Accuracy: 84.00%
Epoch: 4 - Minibatch #200 Accuracy: 87.00%
Epoch: 4 - Minibatch #300 Accuracy: 86.00%
Epoch: 4 - Minibatch #400 Accuracy: 78.00%
Epoch: 4 - Minibatch #500 Accuracy: 86.00%
Epoch: 4 - Minibatch #600 Accuracy: 84.00%
Epoch: 4 - Minibatch #700 Accuracy: 88.00%
Epoch: 4 - Minibatch #800 Accuracy: 84.00%
Epoch: 4 - Minibatch #900 Accuracy: 77.00%
Epoch: 4 - Minibatch #1000 Accuracy: 79.00%
Epoch: 4 - Minibatch #1100 Accuracy: 83.00%
Epoch: 4 - Minibatch #1200 Accuracy: 82.00%
Epoch: 4 - Minibatch #1300 Accuracy: 85.00%
Epoch: 4 - Minibatch #1400 Accuracy: 84.00%
Finished processing epoch: 4
```

```
Train Accuracy: 83.66%
Test Accuracy: 84.08%
Val Accuracy: 83.96%
Epoch: 5 - Minibatch #100 Accuracy: 88.00%
Epoch: 5 - Minibatch #200 Accuracy: 85.00%
Epoch: 5 - Minibatch #300 Accuracy: 89.00%
Epoch: 5 - Minibatch #400 Accuracy: 84.00%
Epoch: 5 - Minibatch #500 Accuracy: 85.00%
Epoch: 5 - Minibatch #600 Accuracy: 89.00%
Epoch: 5 - Minibatch #700 Accuracy: 91.00%
Epoch: 5 - Minibatch #800 Accuracy: 86.00%
Epoch: 5 - Minibatch #900 Accuracy: 79.00%
Epoch: 5 - Minibatch #1000 Accuracy: 81.00%
Epoch: 5 - Minibatch #1100 Accuracy: 87.00%
Epoch: 5 - Minibatch #1200 Accuracy: 83.00%
Epoch: 5 - Minibatch #1300 Accuracy: 85.00%
Epoch: 5 - Minibatch #1400 Accuracy: 83.00%
Finished processing epoch: 5
Train Accuracy: 85.55%
Test Accuracy: 85.23%
Val Accuracy: 84.72%
Epoch: 6 - Minibatch #100 Accuracy: 86.00%
Epoch: 6 - Minibatch #200 Accuracy: 91.00%
Epoch: 6 - Minibatch #300 Accuracy: 90.00%
Epoch: 6 - Minibatch #400 Accuracy: 84.00%
Epoch: 6 - Minibatch #500 Accuracy: 89.00%
Epoch: 6 - Minibatch #600 Accuracy: 93.00%
Epoch: 6 - Minibatch #700 Accuracy: 92.00%
Epoch: 6 - Minibatch #800 Accuracy: 88.00%
Epoch: 6 - Minibatch #900 Accuracy: 83.00%
Epoch: 6 - Minibatch #1000 Accuracy: 81.00%
Epoch: 6 - Minibatch #1100 Accuracy: 83.00%
Epoch: 6 - Minibatch #1200 Accuracy: 86.00%
Epoch: 6 - Minibatch #1300 Accuracy: 90.00%
Epoch: 6 - Minibatch #1400 Accuracy: 86.00%
Finished processing epoch: 6
Train Accuracy: 86.88%
Test Accuracy: 85.69%
Val Accuracy: 84.97%
Epoch: 7 - Minibatch #100 Accuracy: 90.00%
Epoch: 7 - Minibatch #200 Accuracy: 90.00%
Epoch: 7 - Minibatch #300 Accuracy: 93.00%
Epoch: 7 - Minibatch #400 Accuracy: 84.00%
Epoch: 7 - Minibatch #500 Accuracy: 93.00%
Epoch: 7 - Minibatch #600 Accuracy: 95.00%
```

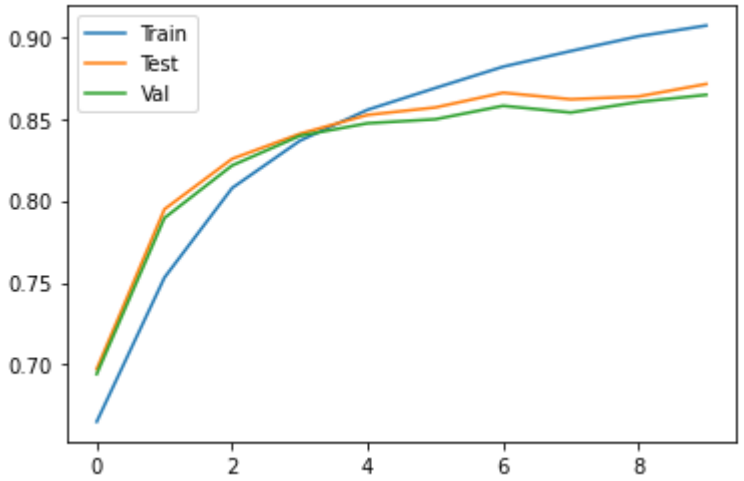
```
Epoch: 7 - Minibatch #700 Accuracy: 93.00%
Epoch: 7 - Minibatch #800 Accuracy: 87.00%
Epoch: 7 - Minibatch #900 Accuracy: 83.00%
Epoch: 7 - Minibatch #1000 Accuracy: 89.00%
Epoch: 7 - Minibatch #1100 Accuracy: 88.00%
Epoch: 7 - Minibatch #1200 Accuracy: 89.00%
Epoch: 7 - Minibatch #1300 Accuracy: 89.00%
Epoch: 7 - Minibatch #1400 Accuracy: 89.00%
Finished processing epoch: 7
Train Accuracy: 88.18%
Test Accuracy: 86.59%
Val Accuracy: 85.79%
Epoch: 8 - Minibatch #100 Accuracy: 91.00%
Epoch: 8 - Minibatch #200 Accuracy: 90.00%
Epoch: 8 - Minibatch #300 Accuracy: 90.00%
Epoch: 8 - Minibatch #400 Accuracy: 90.00%
Epoch: 8 - Minibatch #500 Accuracy: 93.00%
Epoch: 8 - Minibatch #600 Accuracy: 97.00%
Epoch: 8 - Minibatch #700 Accuracy: 96.00%
Epoch: 8 - Minibatch #800 Accuracy: 89.00%
Epoch: 8 - Minibatch #900 Accuracy: 86.00%
Epoch: 8 - Minibatch #1000 Accuracy: 83.00%
Epoch: 8 - Minibatch #1100 Accuracy: 91.00%
Epoch: 8 - Minibatch #1200 Accuracy: 89.00%
Epoch: 8 - Minibatch #1300 Accuracy: 92.00%
Epoch: 8 - Minibatch #1400 Accuracy: 92.00%
Finished processing epoch: 8
Train Accuracy: 89.13%
Test Accuracy: 86.19%
Val Accuracy: 85.38%
Epoch: 9 - Minibatch #100 Accuracy: 91.00%
Epoch: 9 - Minibatch #200 Accuracy: 89.00%
Epoch: 9 - Minibatch #300 Accuracy: 92.00%
Epoch: 9 - Minibatch #400 Accuracy: 89.00%
Epoch: 9 - Minibatch #500 Accuracy: 95.00%
Epoch: 9 - Minibatch #600 Accuracy: 96.00%
Epoch: 9 - Minibatch #700 Accuracy: 98.00%
Epoch: 9 - Minibatch #800 Accuracy: 90.00%
Epoch: 9 - Minibatch #900 Accuracy: 88.00%
Epoch: 9 - Minibatch #1000 Accuracy: 89.00%
Epoch: 9 - Minibatch #1100 Accuracy: 87.00%
Epoch: 9 - Minibatch #1200 Accuracy: 88.00%
Epoch: 9 - Minibatch #1300 Accuracy: 93.00%
Epoch: 9 - Minibatch #1400 Accuracy: 92.00%
Finished processing epoch: 9
```

```
Train Accuracy: 90.03%
Test Accuracy: 86.36%
Val Accuracy: 86.02%
Epoch: 10 - Minibatch #100 Accuracy: 94.00%
Epoch: 10 - Minibatch #200 Accuracy: 93.00%
Epoch: 10 - Minibatch #300 Accuracy: 92.00%
Epoch: 10 - Minibatch #400 Accuracy: 88.00%
Epoch: 10 - Minibatch #500 Accuracy: 91.00%
Epoch: 10 - Minibatch #600 Accuracy: 95.00%
Epoch: 10 - Minibatch #700 Accuracy: 90.00%
Epoch: 10 - Minibatch #800 Accuracy: 90.00%
Epoch: 10 - Minibatch #900 Accuracy: 91.00%
Epoch: 10 - Minibatch #1000 Accuracy: 89.00%
Epoch: 10 - Minibatch #1100 Accuracy: 89.00%
Epoch: 10 - Minibatch #1200 Accuracy: 86.00%
Epoch: 10 - Minibatch #1300 Accuracy: 89.00%
Epoch: 10 - Minibatch #1400 Accuracy: 94.00%
Finished processing epoch: 10
Train Accuracy: 90.69%
Test Accuracy: 87.12%
Val Accuracy: 86.46%
Done training, saving the model..
```

4. Vizualize the accuracy plot

In [8]:

```
plt.plot(train_tracker, label='Train')
plt.plot(test_tracker, label='Test')
plt.plot(val_tracker, label='Val')
plt.legend()
plt.show()
```

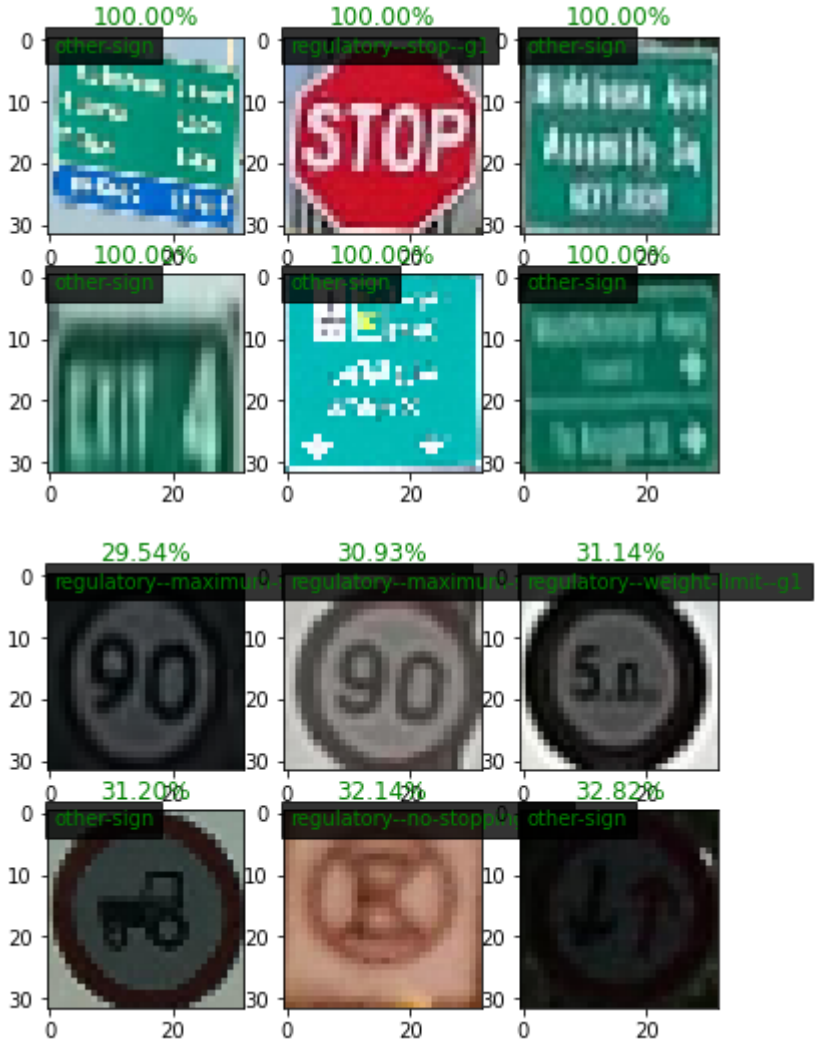


5. Vizualize the Results

In [9]:
```
# load the latest model
test_dataset = MappilaryDataset('data/test.pkl')
model = torch.jit.load('models/latest_model.pt')

# put in eval mode
model.eval();
```

In [10]:
```
# show correct predictions
show_predictions(model, test_dataset, correctly_predicted=True)
```

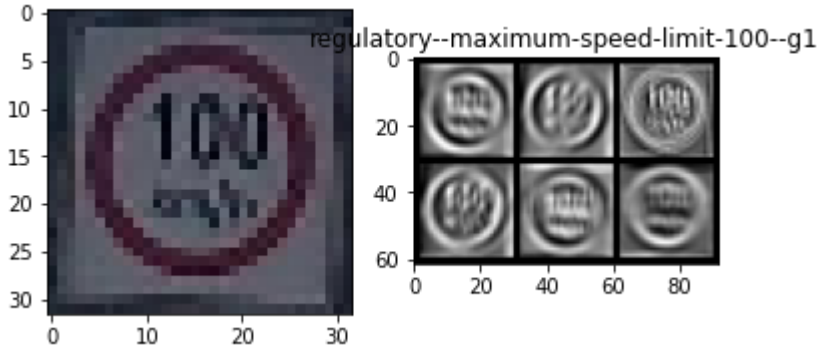


In [11]:

```
# show incorrect predictions
show_predictions(model, test_dataset, correctly_predicted=False)
```

In [13]:

```
# first layer weights for a label
show_weights_for_class(model, test_dataset, label="regulatory--maximum-speed-limit-100--g1")
```


regulatory--maximum-speed-limit-100--g1