Author: Atharva Joshi
Subject: Denoising Autoencoder

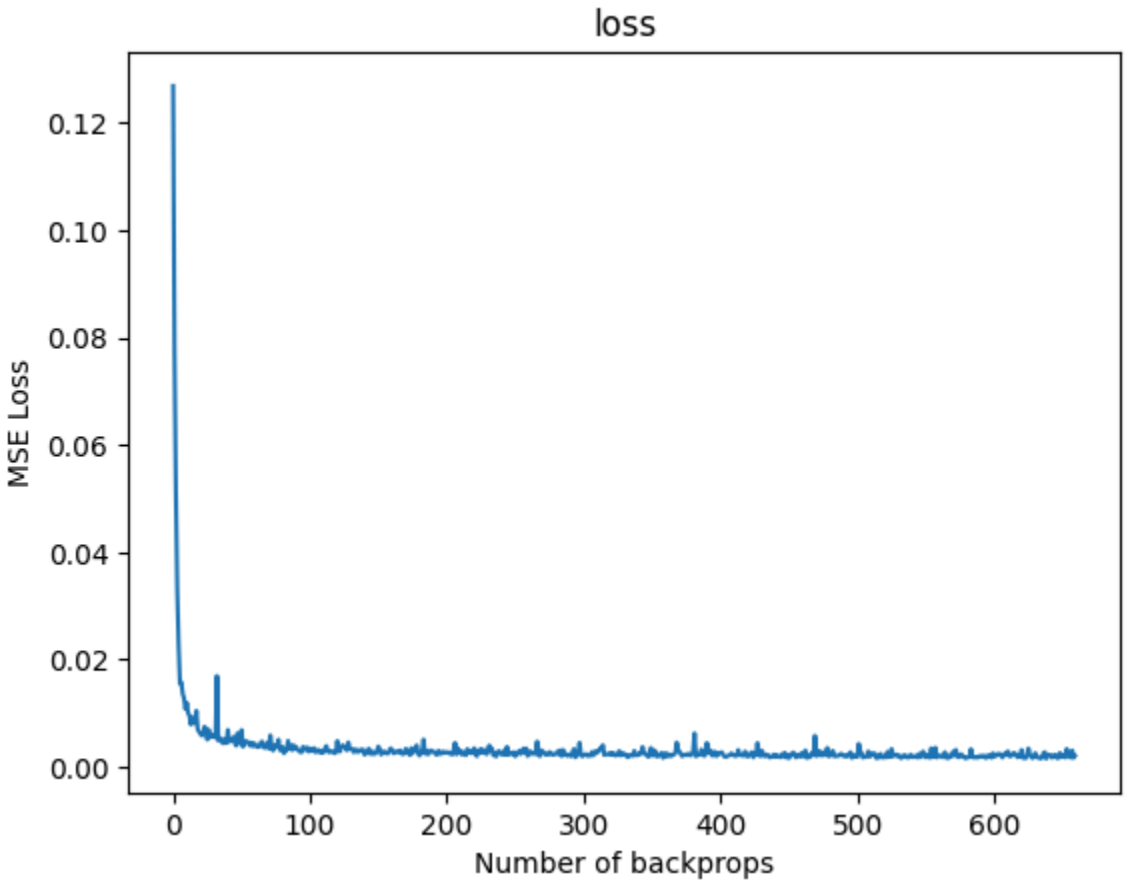
Implementation on FFHQ dataset:
Architecture:

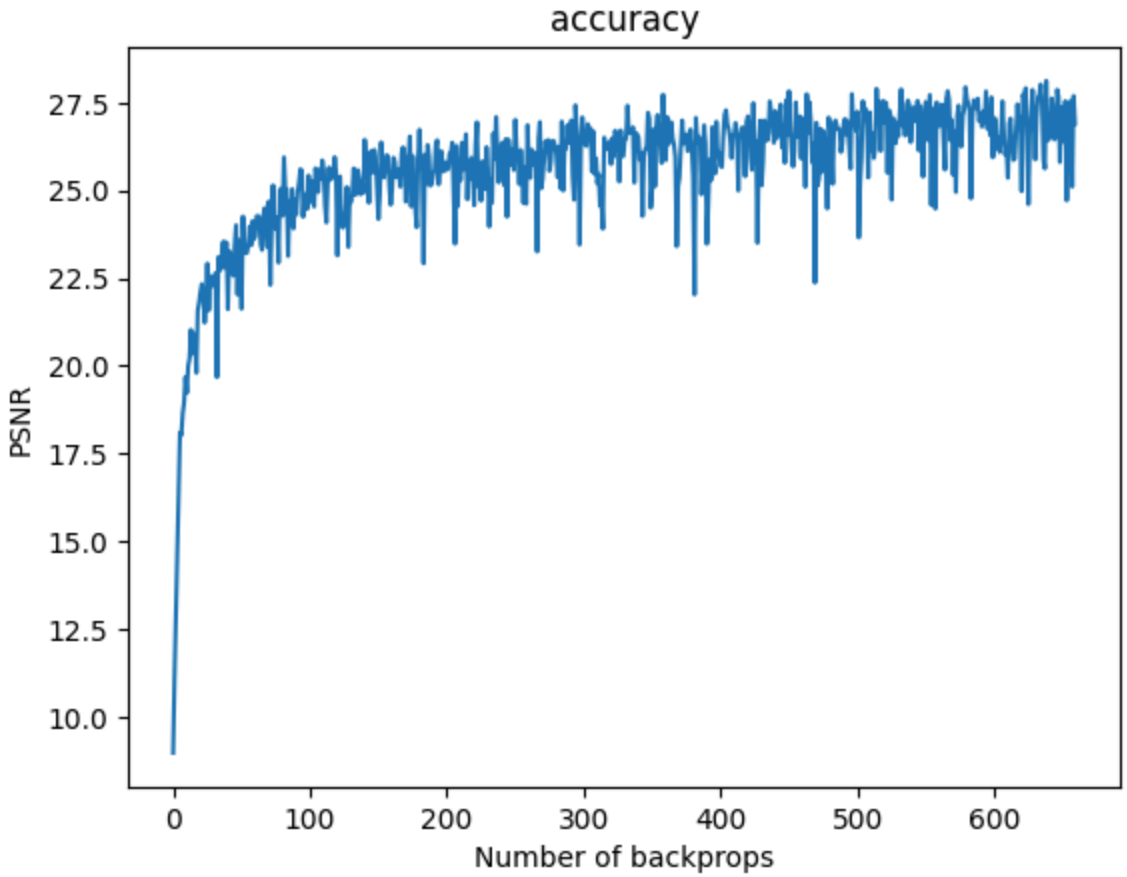
```
class DenoisingAutoencoder(nn.Module):
  def __init__(self):
      super().__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(1, 32, 3, stride=1, padding=1), # (32, 256, 256)
            nn.ReLU(True),
            nn.BatchNorm2d(32),
            nn.MaxPool2d(2, stride=2), # (32, 128, 128)

            nn.Conv2d(32, 64, 3, stride=1, padding=1), # (64, 128, 128)
            nn.ReLU(True),
            nn.MaxPool2d(2, stride=2), # (64, 64, 64)

            nn.Conv2d(64, 128, 3, stride=1, padding=1), # (128, 64, 64)
            nn.ReLU(True),
            nn.BatchNorm2d(128),
            nn.MaxPool2d(2, stride=2) # (128, 32, 32)


        )

        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(128, 64, 3, stride=2, padding=1,
output_padding=1), # (64, 64, 64)
            nn.ReLU(True),

            nn.ConvTranspose2d(64, 32, 3, stride=2, padding=1,
output_padding=1), # (32, 128, 128)
            nn.ReLU(True),
            nn.BatchNorm2d(32),

            nn.ConvTranspose2d(32, 1, 3, stride=2, padding=1,
output_padding=1), # (1, 256, 256)
            nn.Sigmoid()
        )
```

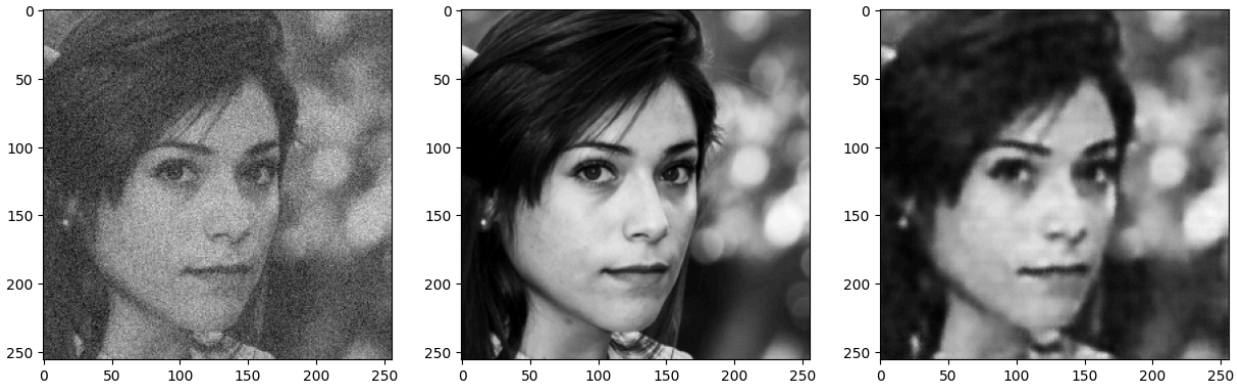
```
def forward(self, x):
    x = self.encoder(x)
    x = self.decoder(x)
    return x
```

Results:

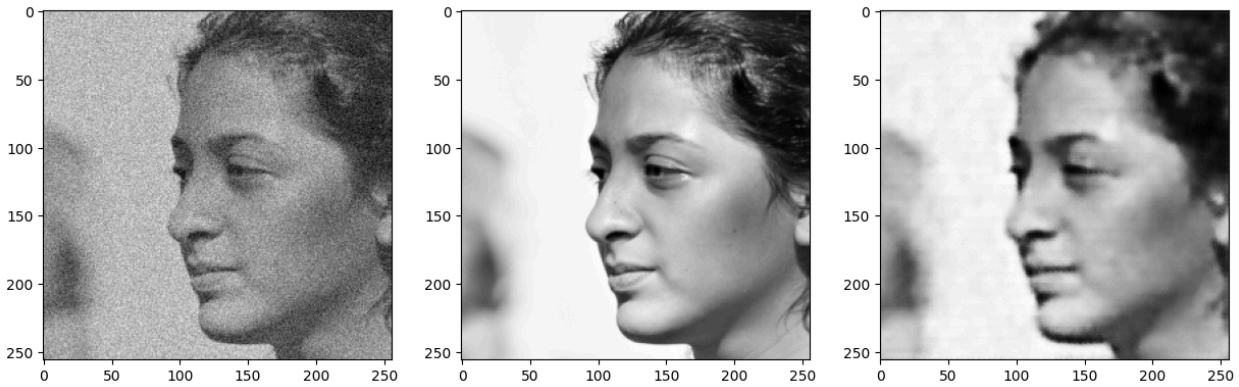
Test accuracy: 28 dB



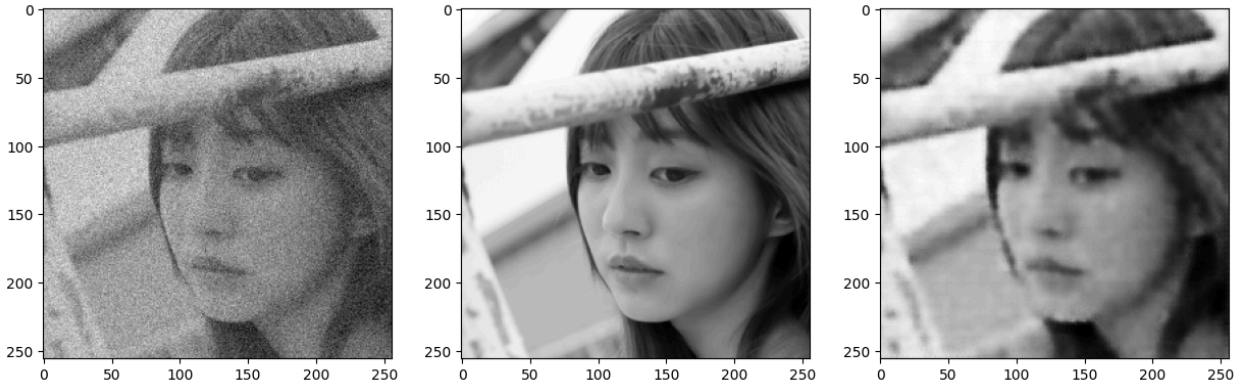




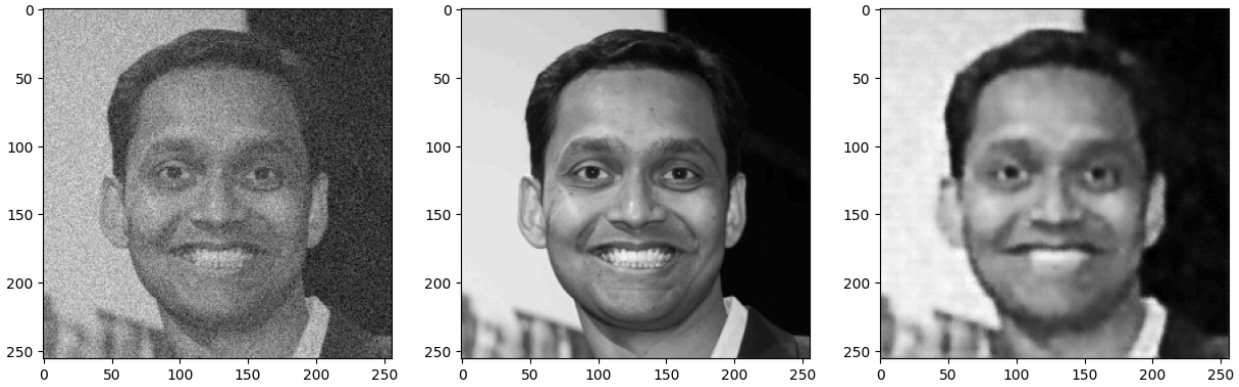
accuracy: 29 dB



accuracy: 28 dB



accuracy: 29 dB



accuracy: 29 dB

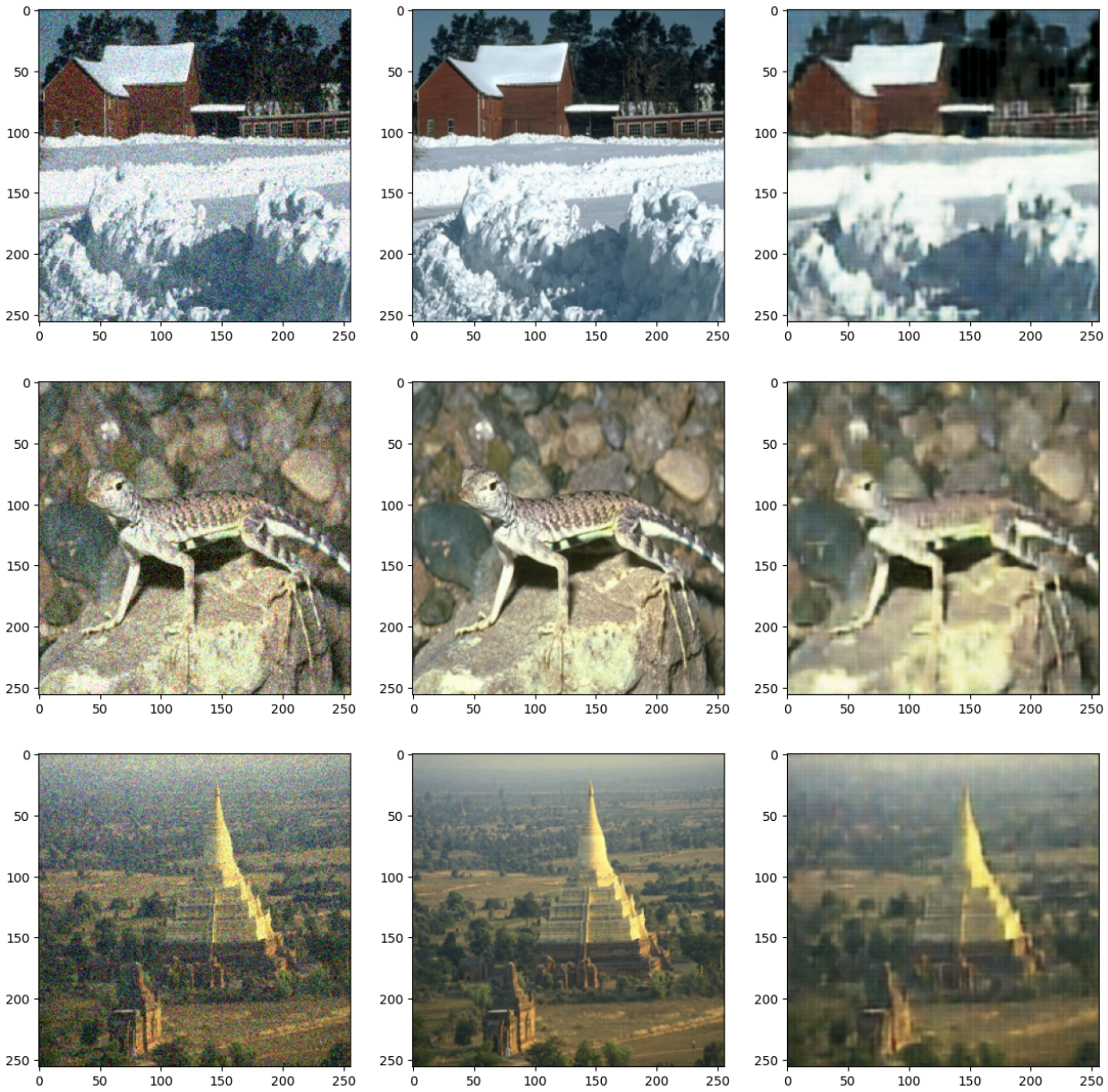
Implementation on BSD100 dataset:
Dataset used: BSD100
Average test PSNR: 25 dB

Architecture:

```python
class DenoisingAutoencoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(3, 32, 3, stride=1, padding=1), # (3, 256, 256)
            nn.ReLU(True),
            nn.BatchNorm2d(32),
            nn.MaxPool2d(2, stride=2), # (32, 128, 128)

            nn.Conv2d(32, 64, 3, stride=1, padding=1), # (64, 128, 128)
            nn.ReLU(True),
            nn.MaxPool2d(2, stride=2), # (64, 64, 64)

            nn.Conv2d(64, 128, 3, stride=1, padding=1), # (128, 64, 64)
            nn.ReLU(True),
            nn.BatchNorm2d(128),
            nn.MaxPool2d(2, stride=2) # (128, 32, 32)


        )
```

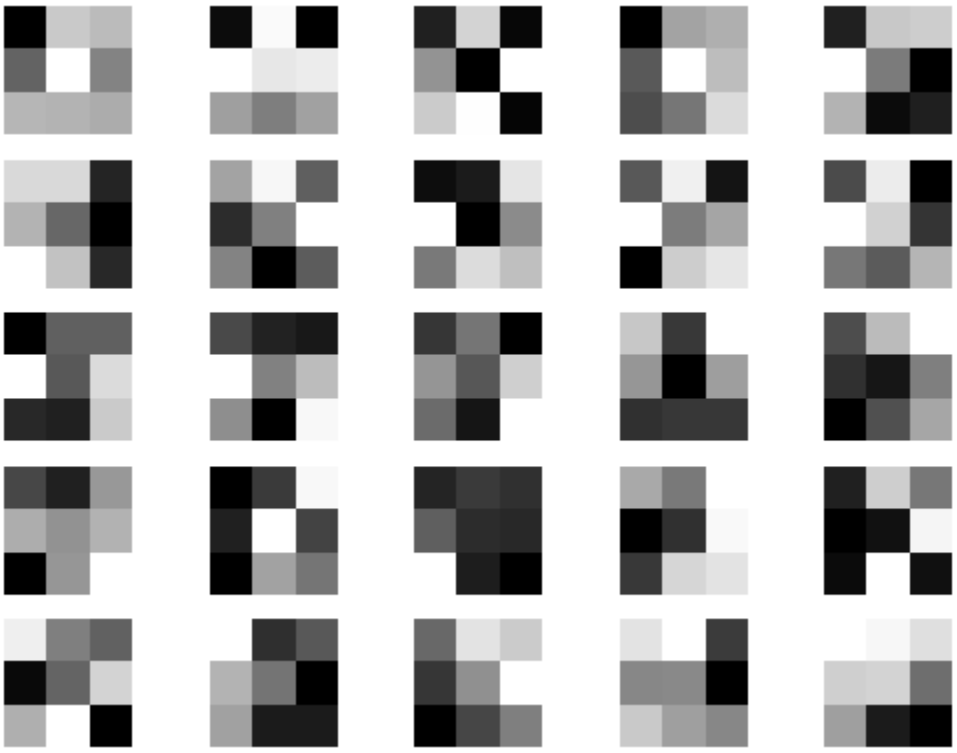
```python
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(128, 64, 3, stride=2, padding=1, output_padding=1), # (64, 64, 64)
            nn.ReLU(True),

            nn.ConvTranspose2d(64, 32, 3, stride=2, padding=1, output_padding=1), # (32, 128, 128)
            nn.ReLU(True),
            nn.BatchNorm2d(32),

            nn.ConvTranspose2d(32, 3, 3, stride=2, padding=1, output_padding=1), # (3, 256, 256)
            nn.Sigmoid()
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

Results

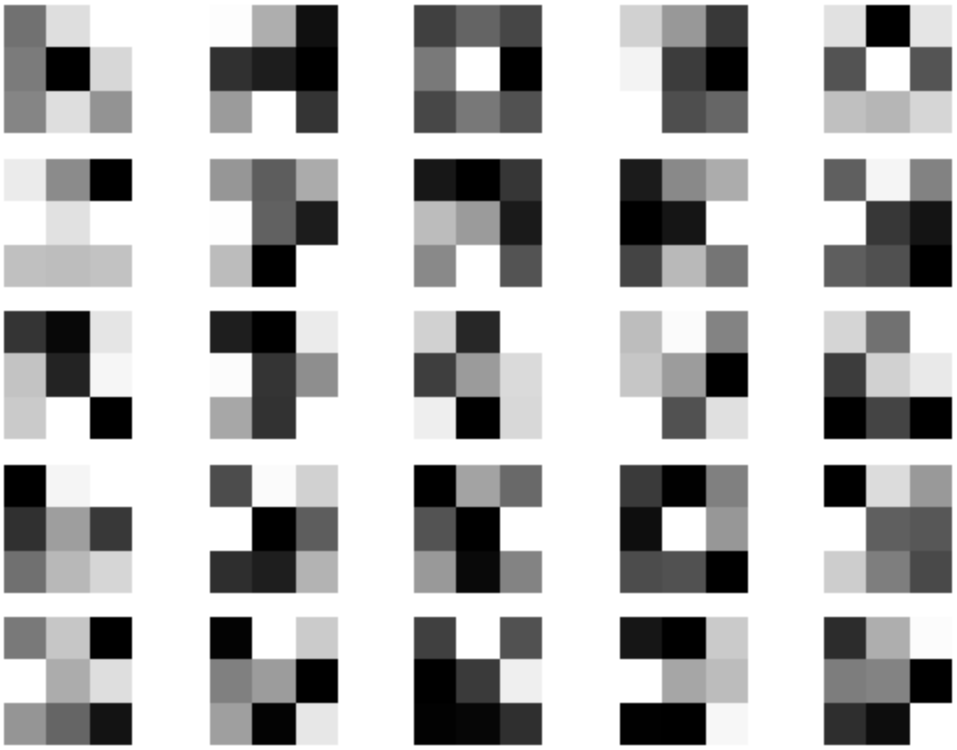


Visualising some kernels:

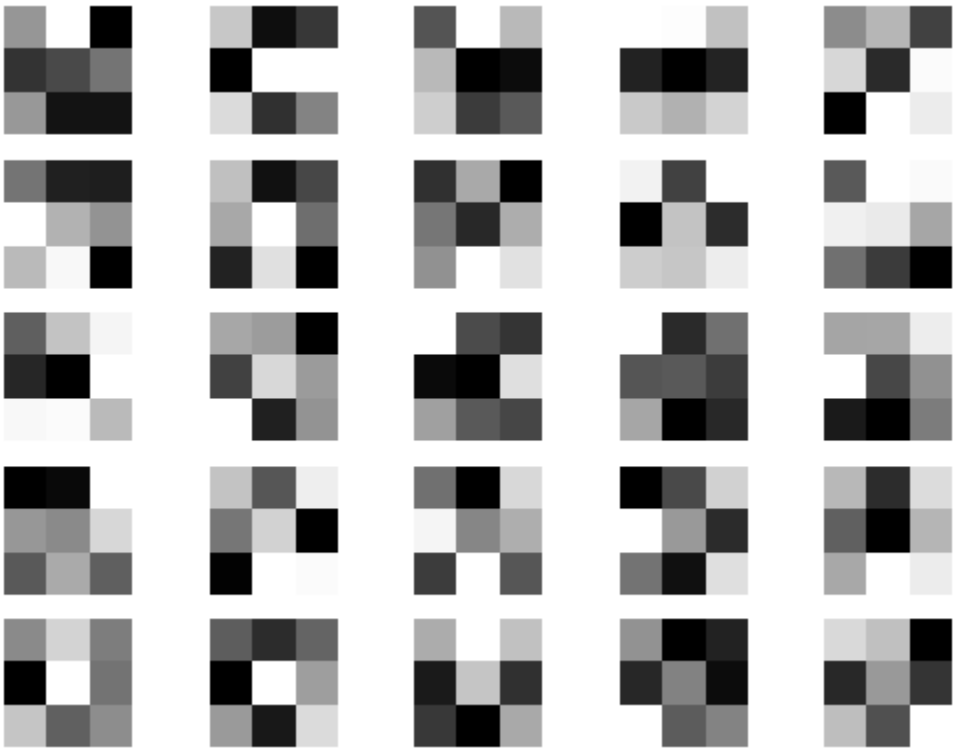
Encoder:
First layer: shape: (32, 3, 3, 3)
Channel 0 (Red channel):



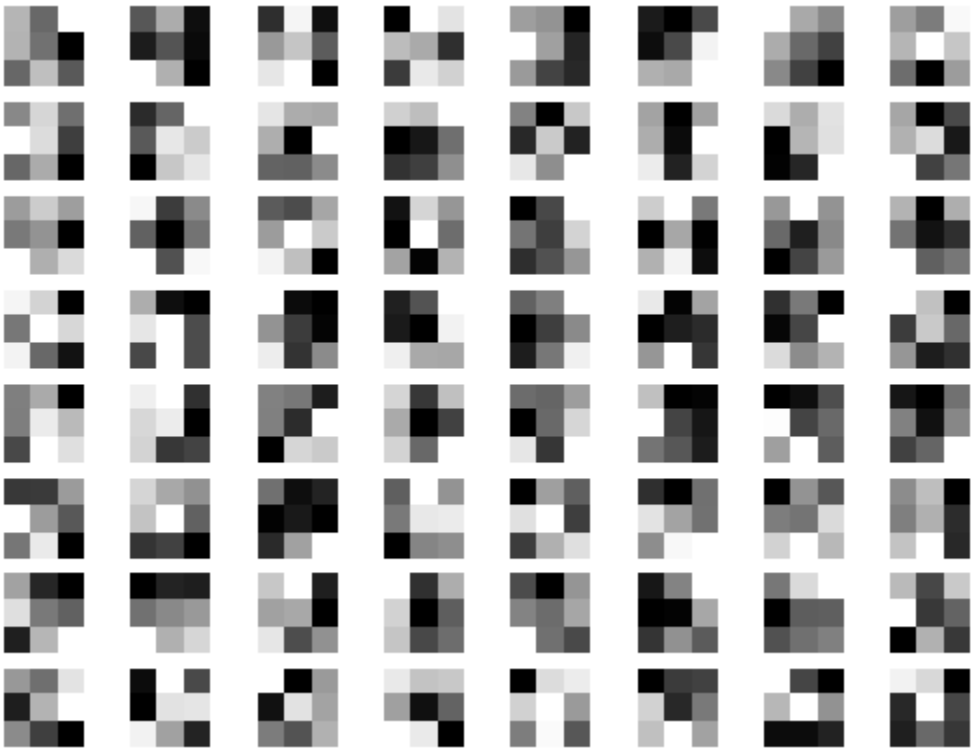
Channel 1 (Green channel):



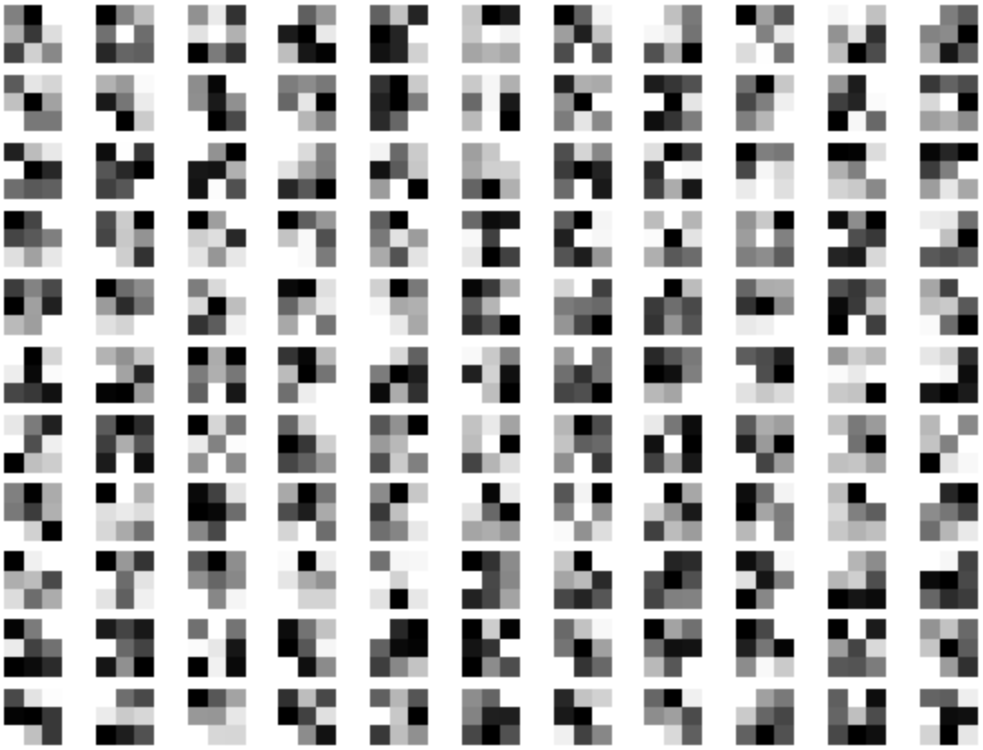
Channel 2 (Blue channel):



Layer 2: shape: (64, 32, 3, 3)
channel 0:

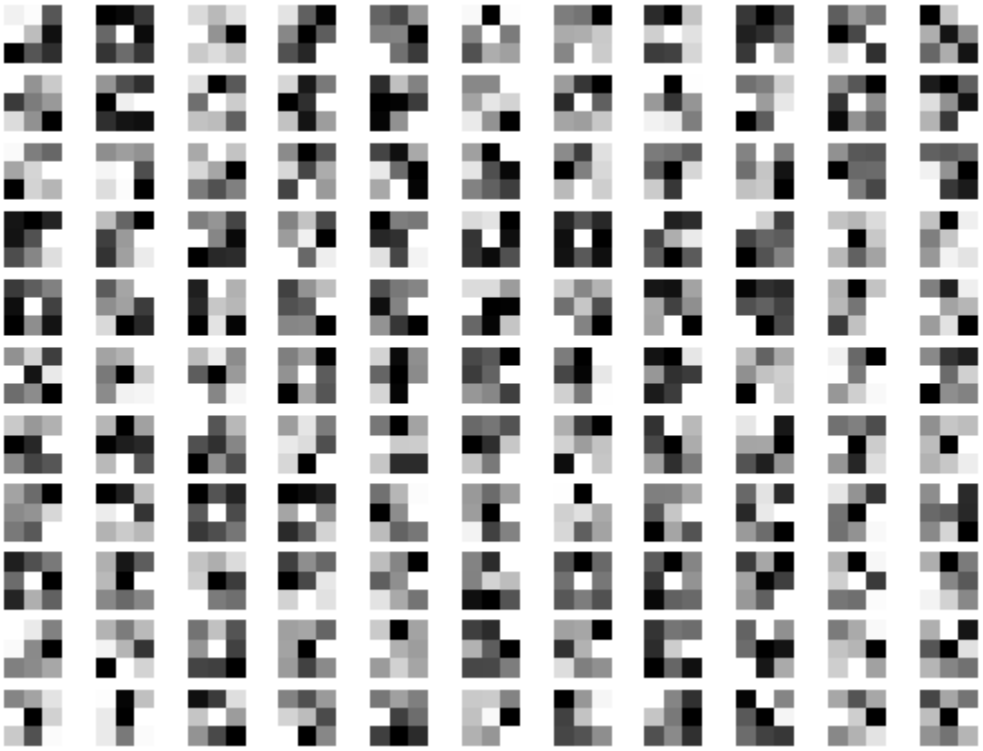


Layer 3: shape: (128, 64, 3, 3)
channel 0:

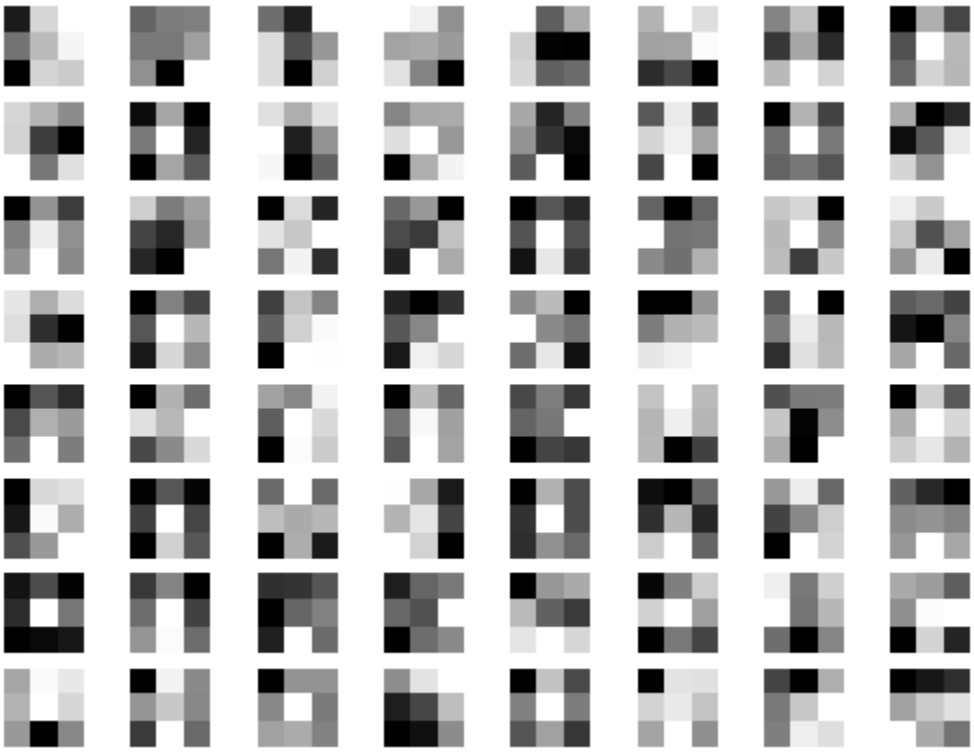


Decoder:

Layer 0:



Layer 1:



Layer 2: