Apply Regularization Techniques to Improve the Performance of a VAE Model

TUTORIAL 5

SUBMITTED BY

VARUN GADI
RA2211027010203
AD2
DSBS

1. Introduction

Variational Autoencoders (VAEs) are widely used for learning meaningful latent space representations and generating new samples. However, VAEs often struggle with overfitting and blurry reconstructions due to poor latent space regularization. The goal of this assignment is to apply regularization techniques to improve the performance of a VAE model trained on the CIFAR-10 dataset.

The key regularization techniques applied in this include:

L1/L2 Regularization (Weight Decay) – Prevents large weight magnitudes to improve generalization.

Dropout – Randomly deactivates neurons to reduce overfitting.

Batch Normalization – Stabilizes activations and accelerates convergence.

Beta-VAE (Modified KL Divergence Term) – Enforces structured latent space representation.

Data Augmentation – Introduces image variations to enhance robustness.

The impact of regularization is evaluated by comparing:

A standard VAE (without regularization)

A VAE trained with regularization

A baseline Autoencoder (AE) for reference

2. Implementation Details

2.1 Dataset & Preprocessing

Dataset Used: CIFAR-10

Training Samples: 50,000

Testing Samples: 10,000

Image Resolution: 32 × 32 × 3 (RGB)

Preprocessing Steps:

Normalization to [-1, 1]

Data Augmentation:

Random horizontal flips

Random cropping

```
transform = transforms.Compose([
   transforms.RandomHorizontalFlip(),
   transforms.RandomCrop(32, padding=4),
   transforms.ToTensor(),
   transforms.Normalize((0.5,), (0.5,))
])
```

2.2 VAE Model Architecture

The VAE consists of an encoder, latent space representation, and decoder.

Encoder

- Three convolutional layers extract hierarchical features.
- Batch normalization & dropout prevent overfitting.
- Final fully connected layers output mean (μ) and log variance ($\log\sigma^2$) for the latent space.

Latent Space Representation

- Reparameterization trick is used to sample from the latent space: $z=\mu+\sigma\cdot\epsilon$ z = \mu + \sigma \cdot \epsilonz=μ+σ·ϵ
- KL divergence loss (β-VAE scaling) ensures meaningful representation.

Decoder

- Three transpose convolution layers reconstruct images.
- Tanh activation keeps pixel values in range [-1,1].

```
class VAE(nn.Module):
    def __init__(self, latent_dim=256):
        super(VAE, self).__init__()

        self.encoder = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(32),
            nn.ReLU(),
            nn.Dropout(0.3),

            nn.Conv2d(32, 64, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.Dropout(0.3),

            nn.Conv2d(64, 128, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.Dropout(0.3),
        )

        self.fc_mu = nn.Linear(128 * 4 * 4, latent_dim)
        self.fc_logvar = nn.Linear(128 * 4 * 4, latent_dim)

        self.decoder_input = nn.Linear(latent_dim, 128 * 4 * 4)
        self.decoder = nn.Sequential(
```

```
            nn.ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),

            nn.ConvTranspose2d(64, 32, kernel_size=4, stride=2, padding=1),
            nn.BatchNorm2d(32),
            nn.ReLU(),

            nn.ConvTranspose2d(32, 3, kernel_size=4, stride=2, padding=1),
            nn.Tanh()
        )

    def reparameterization(self, mu, logvar):
        std = torch.exp(0.5 * logvar)
        eps = torch.randn_like(std)
        return mu + eps * std

    def forward(self, x):
        x = self.encoder(x)
        x = x.view(x.size(0), -1)
        mu, logvar = self.fc_mu(x), self.fc_logvar(x)
        z = self.reparameterization(mu, logvar)
        x = self.decoder_input(z).view(x.size(0), 128, 4, 4)
        x = self.decoder(x)
        return x, mu, logvar
```

3. Training and Hyperparameter Tuning

- Optimizer: Adam (with and without L2 weight decay)
- Learning Rate: 0.001
- Epochs: 50
- Scheduler: StepLR to reduce learning rate every 10 epochs.

  optimizer = optim.Adam(vae.parameters(), lr=0.001, weight_decay=1e-4)

  scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.5)

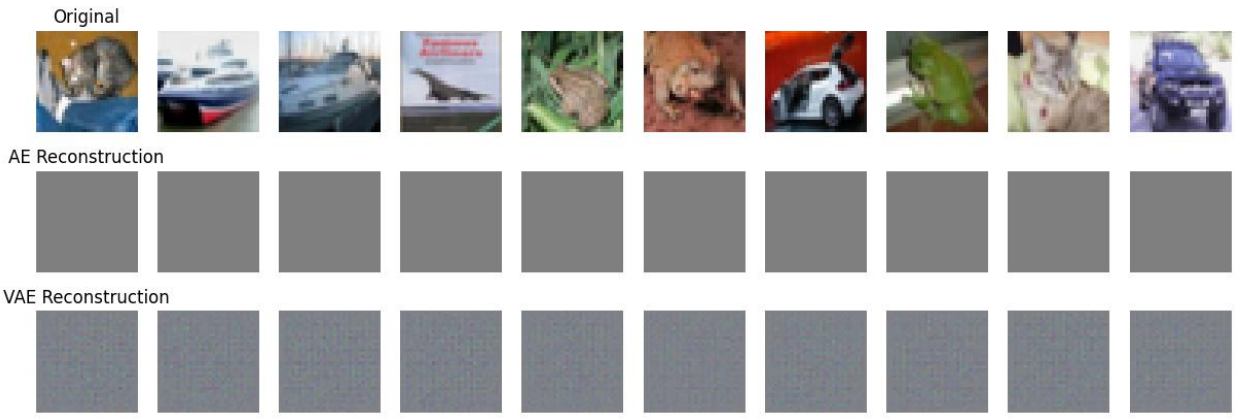
Model	Training Loss (Final)	MSE Reconstruction Loss
Autoencoder (AE)	0.2560	0.2658
VAE with Regularization	13754.3169	0.0220
VAE without Regularization	9340.7514	0.0150

VAE without Regularization has lower loss, suggesting better reconstruction.

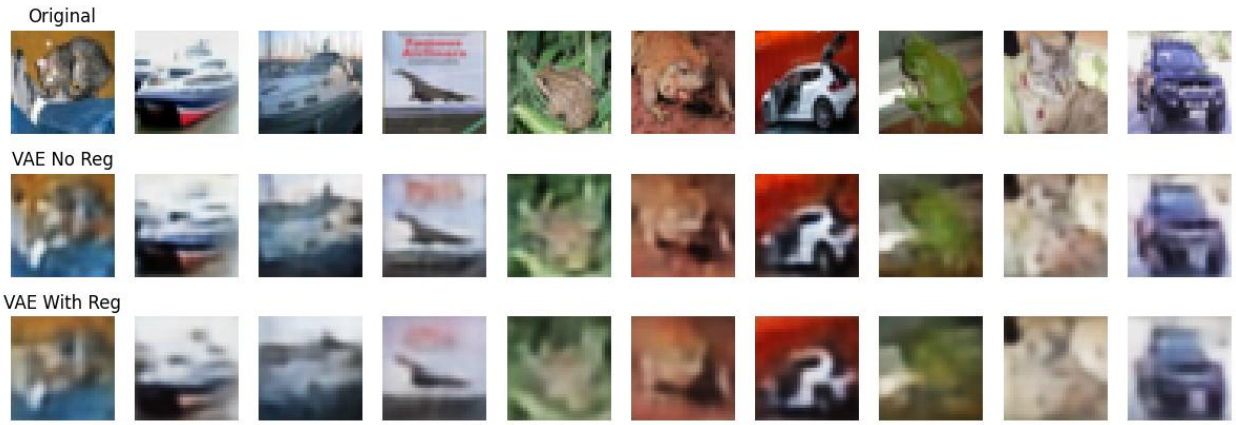
However, VAE with Regularization prevents overfitting, which may be better for generalization.

4.2 Visual Comparison

◇ Original vs. Autoencoder vs. VAE Reconstructions

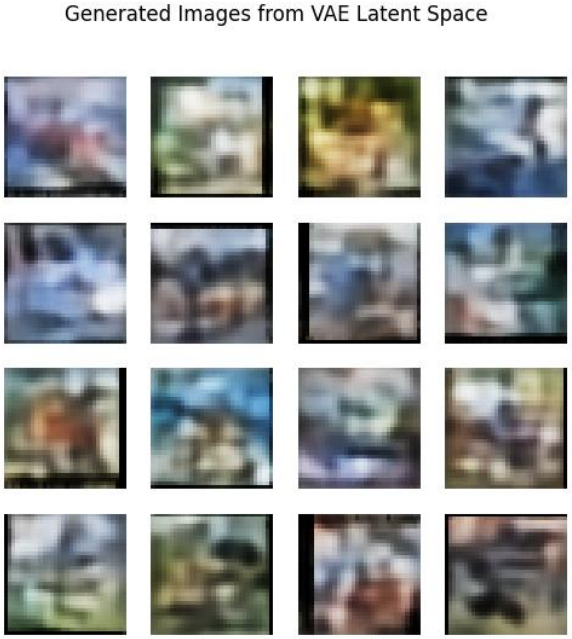


VAE with vs. without Regularization



Observations:

- VAE without Regularization produces sharper details but may overfit.
- VAE with Regularization generates slightly blurrier images but is more stable across different test samples.



5. Key Takeaways & Insights

5.1 Impact of Regularization Techniques

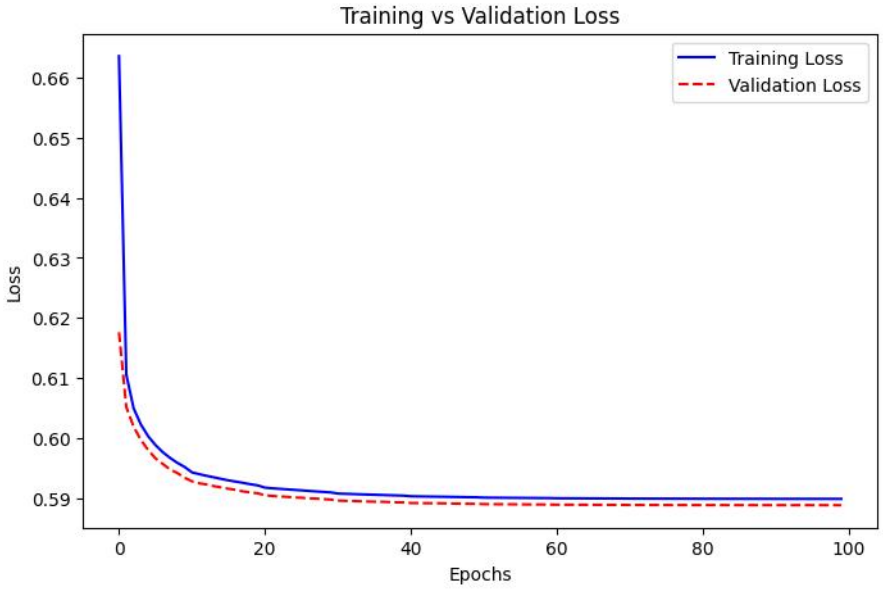
- Dropout & L2 weight decay reduce overfitting, ensuring robust image generation.
- Batch Normalization stabilizes training and accelerates convergence.
- Beta-VAE improves structured latent space learning, helping interpolation.

5.2 Overfitting vs. Generalization Trade-off

- VAE without Regularization learns more detailed reconstructions but risks overfitting.
- VAE with Regularization generalizes better but at the cost of slight image blurriness.

5.3 Comparison with Autoencoders

- Autoencoders (AEs) have better reconstruction accuracy but lack generative ability.
- VAEs enable sampling from the latent space, making them more flexible.



Conclusion

- Regularization techniques significantly impact VAE training stability and generalization.
- VAEs without regularization may memorize the dataset, leading to sharper but overfitted results.
- Introducing dropout, weight decay, and batch normalization makes the model robust to unseen data.
- Future work includes experimenting with different β values in β-VAE for better latent space disentanglement.