Observations

Vivian Sedov

March 14, 2023

Contents

1 Input Parser

Issue : Generator collapses and mode collapse occurs when the auxiliary loss for both hair and eye drops suddenly.

Explanation : The auxiliary loss is acting as a regularizer for the generator, ensuring that it produces realistic images that also match the specified attributes (hair and eye color). When the auxiliary loss drops suddenly, it means that the generator is no longer being regularized properly, and as a result, it may start producing images that are only realistic but do not match the specified attributes, leading to mode collapse.

Solution :

- Increase regularization strength: One way to prevent sudden drops in the auxiliary loss is to increase the strength of the regularization used in the loss function. This can be achieved by increasing the weight of the auxiliary loss term or introducing additional constraints on the generator.
- Add noise to the latent code: Adding noise to the latent code can help regularize the generator and prevent mode collapse. This can be achieved by introducing random perturbations to the input vector z at each training iteration.
- Use different loss functions: Another approach could be to use a different loss function that penalizes the generator for producing images that do not match the specified attributes. For example, one could use a Wasserstein distance-based loss that penalizes the difference between the generated and real data distributions.
- Monitor auxiliary loss during training: Monitoring the auxiliary loss during training and adjusting the regularization strength accordingly can help prevent sudden drops. This can involve setting up early stopping criteria that trigger when the loss drops below a certain threshold or using adaptive regularization techniques that adjust the regularization strength based on the loss value.

- Use adversarial training: Adversarial training is a technique that involves training the generator and discriminator in an alternating fashion. This can help stabilize the training process and prevent mode collapse by encouraging the generator to produce more diverse images.

Implications : This is a common issue in auxiliary generative adversarial networks, and understanding its underlying causes and solutions can help improve the performance and stability of such networks.

1.0.1 Why is this occouring

In the auxiliary generative adversarial network (GAN) framework, the generator is trained to produce images that not only look realistic but also match certain attributes (in this case, hair and eye color). The auxiliary loss function helps achieve this by penalizing the generator for producing images that do not match the specified attributes. Specifically, the hair auxiliary loss and eye auxiliary loss terms measure the difference between the predicted and true hair and eye color labels, respectively.

When the hair and eye auxiliary losses drop suddenly, it means that the generator is no longer being penalized sufficiently for producing images that do not match the specified attributes. This can cause the generator to produce images that look realistic but have incorrect hair and eye colors, leading to mode collapse.

One way to address this issue is to add an extra discriminator layer that specifically targets the attribute labels (i.e., hair and eye color). This can help provide stronger feedback to the generator on whether the generated images match the specified attributes or not. By adding this extra layer, the discriminator can learn to distinguish between images that not only look realistic but also have the correct hair and eye colors.

Here is some example code where the hair and eye auxiliary losses are being calculated:

```
real_hair_aux_loss = criterion(real_hair_predict, hair_tags)
real_eye_aux_loss = criterion(real_eye_predict, eye_tags)
real_classifier_loss = real_hair_aux_loss + real_eye_aux_loss
```

In the above code snippet, real_hair_predict and real_eye_predict are the predicted hair and eye color labels for the real images, respectively, and hair_tags and eye_tags are the true hair and eye color labels, respectively. The auxiliary losses (real_hair_aux_loss and real_eye_aux_loss) are calculated by comparing the predicted labels with the true labels using the binary cross-entropy loss function (criterion).

Mathematically, adding an extra discriminator layer can be seen as introducing an additional term in the objective function that encourages the generator to produce images that match the specified attributes. The discriminator tries to minimize this term by correctly classifying the attribute labels, while the generator tries to maximize it by producing images that fool the discriminator into thinking they have the correct attribute labels. This results in a more robust and stable training process that helps prevent mode collapse.

In summary, sudden drops in the hair and eye auxiliary losses can lead to generator collapse, and adding an extra discriminator layer that specifically targets the attribute labels can help address this issue by providing stronger feedback to the generator on whether the generated images match the specified attributes or not.