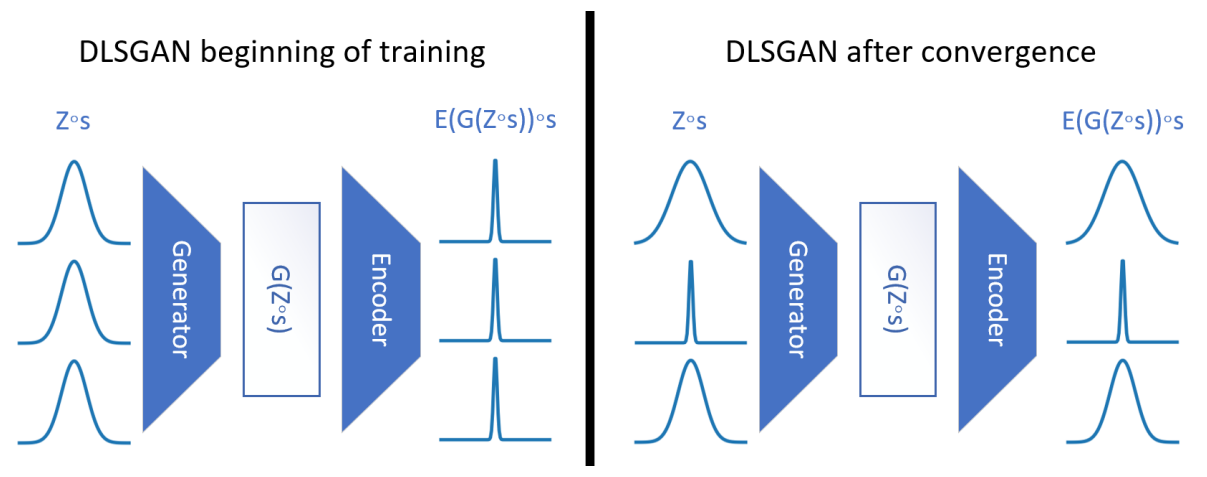
Applications of Dynamic Latent Scale GAN: OOD Detection and Continuous Attribute Editing

Jeongik Cho

1 Out-of-Distribution (OOD) Detection

Out-of-distribution (OOD) detection refers to identifying whether a given input sample comes from the same distribution as the training data (in-distribution), or from a different, unknown distribution (out-of-distribution). Unlike conventional binary classification, OOD detection assumes access only to in-distribution data during training, making generalization to unseen data distributions particularly challenging—especially in high-dimensional input spaces where OOD boundaries are inherently difficult to model.



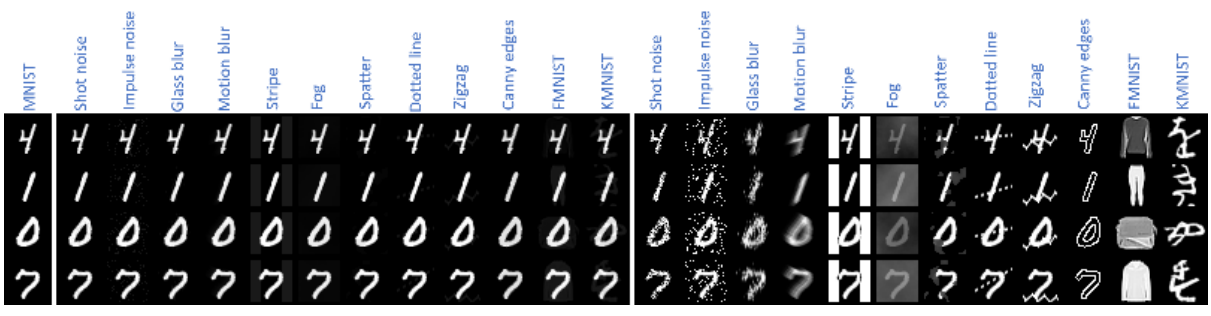
The main features that make DLSGAN suitable for OOD detection are: (1) the latent entropy becomes optimal for representing the data distribution, and (2) each dimension of the latent distribution follows a simple, independent distribution. These features allow DLSGAN's encoder to be utilized for likelihood-based OOD detection. The figure above shows that, after DLSGAN has converged, the scaled latent distribution has the appropriate entropy to represent the data distribution.

The following equation defines the OOD score based on the encoder output:

$$OOD\ score = -\sum_{i=1}^{d_z} \log f(E_i(x)|\mu_i, v_i) \tag{1}$$

In the formula, $E_i(x)$ denotes the i-th element of predicted latent vector from the encoder, while μ_i and v_i are the mean and variance of the i-th element of the encoder output distribution. An input sample is classified as OOD if its score exceeds a predefined threshold.

To validate this approach, we conducted experiments on the MNIST dataset. The figure below shows in-distribution (ID) samples in the first column, followed by near OOD (columns 2-13) and far OOD (columns 14-25) samples. Near OOD samples are particularly challenging since they closely resemble the ID samples.



In the experiments, all models are trained solely on the ID training data, and evaluation is performed using ID test data and OOD data. The table below summarizes the performance:

AUROC (×100)		GAN						Auto-encoder	Classifier							
		DLSGAN [51]		InfoGAN [39]		MSEGAN		Reconst-ruction [65]	Energy score [57] with ReAct [58]							
		NLP (ours)	Rec	NLP	Rec	NLP	Rec	Rec	t=1.0 p=0.85	t=1.0 p=0.90	t=1.0 p=0.95	t=1.0 p=1.0	t=10.0 p=0.85	t=10.0 p=0.90	t=10.0 p=0.95	t=10.0 p=1.0
Near OOD	Shot noise	50.68	50.15	50.68	48.96	50.62	49.17	52.85	50.61	50.98	51.40	51.94	50.50	50.92	51.37	51.93
	Impulse noise	93.47	65.74	57.07	52.88	59.35	53.50	72.20	50.64	51.00	51.41	51.92	50.54	50.94	51.38	51.91
	Glass blur	66.60	57.45	52.58	47.40	52.53	48.00	47.96	50.90	51.85	52.95	54.28	50.64	51.70	52.88	54.26
	Motion blur	85.13	65.14	55.84	48.65	55.94	49.33	50.05	52.08	53.05	54.10	55.31	51.79	52.89	54.03	55.28
	Stripe	100.00	95.27	97.17	82.38	97.61	79.36	98.72	51.12	51.65	52.37	53.36	51.12	51.67	52.41	53.41
	Fog	99.44	89.74	67.29	53.54	68.34	54.37	74.53	52.21	53.74	55.45	57.39	51.84	53.54	55.36	57.37
	Spatter	70.81	64.16	52.48	49.83	52.74	49.99	52.10	50.34	50.54	50.76	51.05	50.29	50.51	50.75	51.05
	Dotted line	72.04	59.99	53.59	52.64	54.91	53.04	56.05	50.22	50.29	50.36	50.46	50.18	50.26	50.35	50.45
	Zigzag	90.99	73.19	61.02	54.82	63.77	55.89	64.13	50.44	50.58	50.74	50.93	50.37	50.54	50.72	50.92
	Canny edges	87.58	60.47	57.53	48.39	57.08	49.26	62.08	52.57	54.54	56.71	59.19	52.00	54.21	56.56	59.12
	FMNIST	99.85	87.92	73.30	55.19	73.78	57.78	80.65	53.57	55.37	57.33	59.56	52.95	55.01	57.15	59.48
	KMNIST	99.46	84.34	71.96	54.96	72.98	56.90	82.90	54.11	55.93	57.93	60.20	53.46	55.55	57.74	60.10
Far OOD	Shot noise	94.97	71.55	69.67	59.81	68.46	61.23	99.74	69.49	71.23	73.06	75.03	67.07	69.75	72.12	74.34
	Impulse noise	100.00	99.89	99.99	99.84	100.00	99.83	100.00	83.69	83.55	83.27	83.22	81.03	81.77	81.94	82.03
	Glass blur	99.89	72.23	93.39	58.85	93.48	60.84	99.80	82.57	86.73	89.95	91.75	78.08	84.34	88.60	90.76
	Motion blur	100.00	80.51	96.73	59.94	97.57	65.29	97.60	83.49	86.59	88.88	90.27	80.59	85.11	88.07	89.68
	Stripe	100.00	100.00	100.00	100.00	100.00	100.00	100.00	92.99	92.86	92.25	91.36	93.26	93.19	92.52	91.52
	Fog	100.00	97.85	99.97	96.78	99.98	95.57	99.99	99.27	99.65	99.79	99.82	96.50	98.53	99.22	99.40
	Spatter	99.54	87.41	89.73	75.56	91.11	75.96	99.41	63.21	64.48	65.69	66.90	61.82	63.60	65.12	66.47
	Dotted line	99.41	89.91	86.53	88.04	87.61	87.88	99.96	64.04	63.51	62.79	62.03	61.74	61.88	61.55	60.95
	Zigzag	99.89	95.12	94.27	95.34	95.01	95.14	99.99	71.91	71.01	69.63	67.84	68.53	68.46	67.53	65.93
	Canny edges	100.00	94.04	99.93	88.27	99.92	89.35	99.94	92.41	89.76	84.03	65.95	90.17	87.62	81.34	62.08
	FMNIST	100.00	97.64	99.96	93.96	99.98	94.01	99.98	96.37	97.06	97.46	97.60	94.20	95.87	96.72	97.00
	KMNIST	99.94	97.49	98.02	95.89	98.22	96.55	99.98	94.89	94.81	94.32	93.49	92.29	93.04	92.87	92.01

As shown, DLSGAN (NLP) significantly outperforms existing methods, especially in near OOD detection, which is more difficult.

2 Continuous Attribute Editing

The second application focuses on continuous attribute manipulation using DLSGAN. InterFaceGAN has demonstrated that discrete attributes can be linearly separated in the GAN latent space. By combining this idea with generator inversion via DLSGAN, we can continuously edit semantic attributes of input images.

The core idea is to train a class-conditional DLSGAN that aligns with a fixed linear classifier in the latent space. This classifier is initialized with random weights and kept fixed throughout training. During training, it is used to generate pseudo labels for each latent vector, which are then used to train the class-conditional DLSGAN. The following equation defines the pseudo label assigned to each latent vector during training.

$$c_f = argmax\ onehot((z \circ s)w + b) \tag{2}$$

In this formulation, "z" is the latent vector, "s" is the scale vector, "w" and "b" are the weight and bias of the fixed linear classifier, and "c_f" is the pseudo label. The "argmax onehot" function returns a one-hot vector whose nonzero element corresponds to the index of the largest dimension of the classifier output. These pseudo labels are then used to train a class-conditional DLSGAN along with the real labels of the input data.

Once the model is trained, we can manipulate the predicted latent vector of an input sample by gradually modifying the output of the fixed linear classifier. This continuous adjustment in latent space results in smooth and semantically meaningful changes in the generated images.

In the figure above, the first column shows the original input image, the second column shows its reconstruction, and the remaining columns show progressive transformations in the "gender" attribute. The transitions are smooth and continuous, made possible by the structured latent space and effective inversion mechanism provided by DLSGAN.

You can find the full implementation of both experiments at:

`https://github.com/jeongik-jo/AnoDLSGAN`

`https://github.com/jeongik-jo/AEDLSGAN`