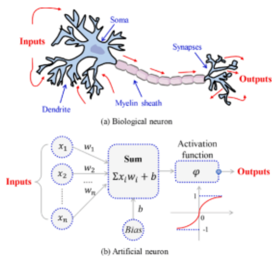
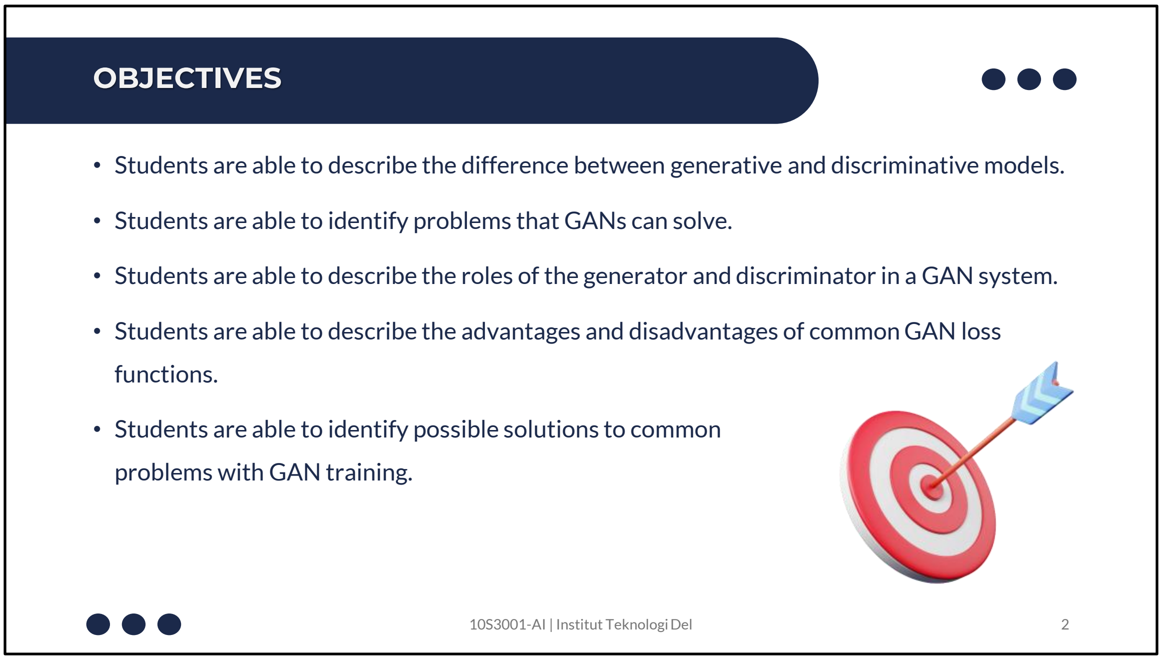
GENERATIVE ADVERSARIAL NETWORKS

10S3001 – Artificial Intelligence

by Samuel I. G. Situmeang



Faculty of Informatics and Electrical Engineering

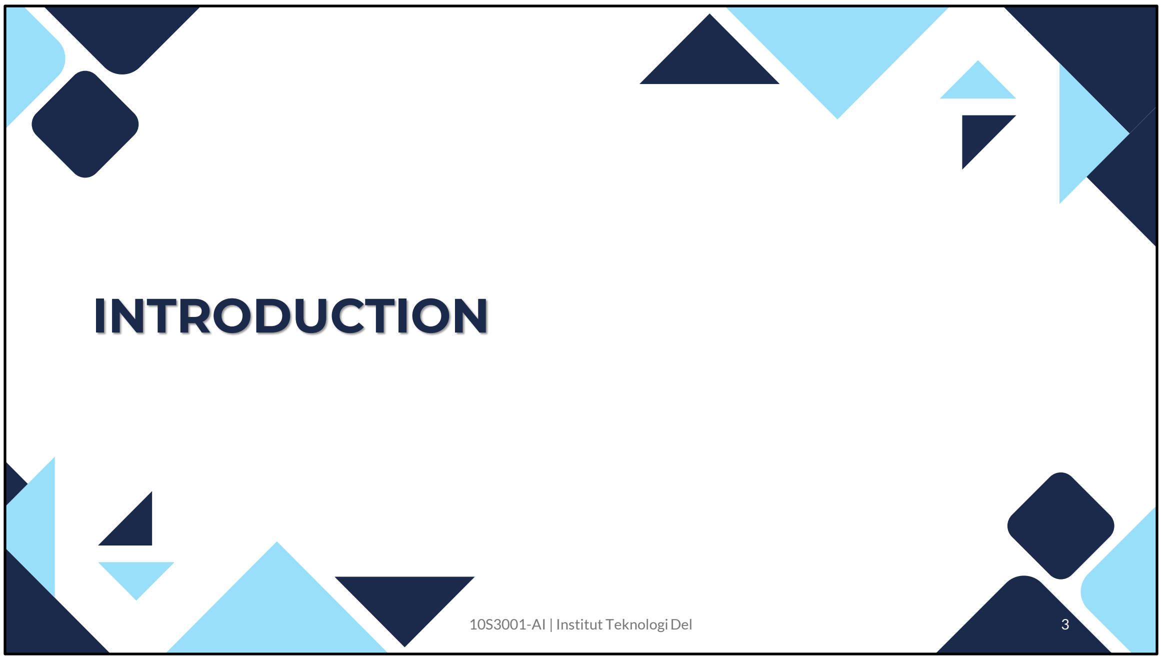


OBJECTIVES

- Students are able to describe the difference between generative and discriminative models.
- Students are able to identify problems that GANs can solve.
- Students are able to describe the roles of the generator and discriminator in a GAN system.
- Students are able to describe the advantages and disadvantages of common GAN loss functions.
- Students are able to identify possible solutions to common problems with GAN training.

10S3001-AI | Institut Teknologi Del 2

- Mahasiswa mampu menjelaskan perbedaan antara model generatif dan diskriminatif.
- Mahasiswa mampu mengidentifikasi masalah yang dapat dipecahkan oleh GAN.
- Mahasiswa mampu menjelaskan peran generator dan diskriminator dalam sistem GAN.
- Mahasiswa mampu menjelaskan kelebihan dan kekurangan fungsi kerugian GAN yang umum.
- Mahasiswa mampu mengidentifikasi kemungkinan solusi untuk masalah umum dengan pelatihan GAN.



INTRODUCTION

WHAT IS GAN?

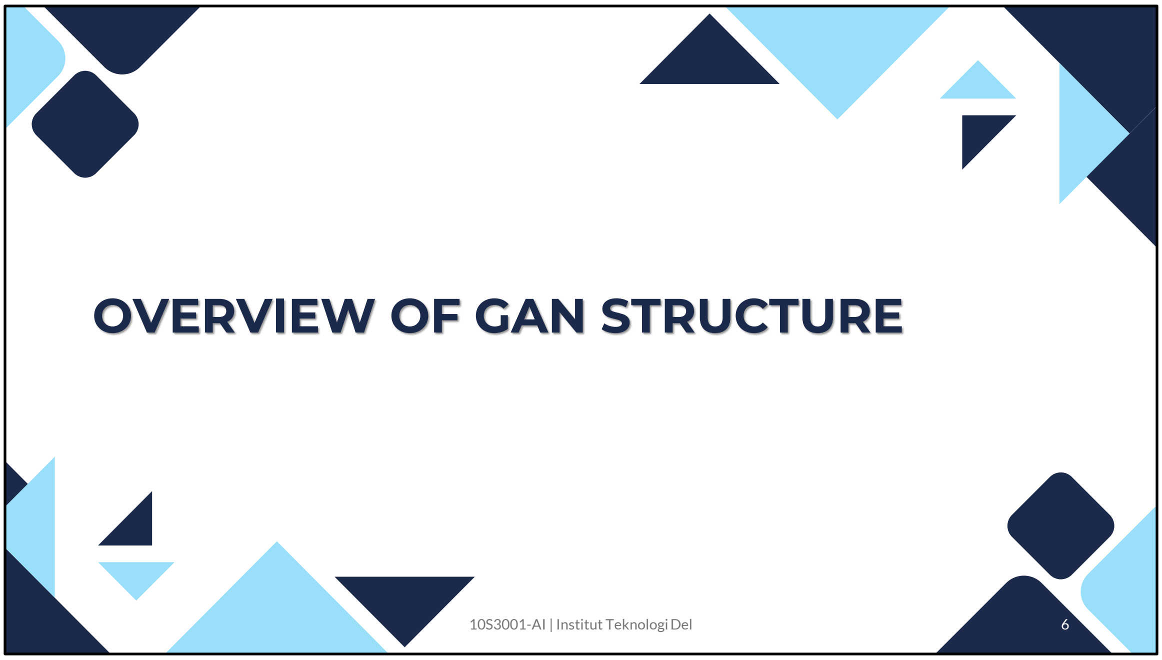
- GANs are generative models: they create new data instances that resemble your training data.
  - For example, GANs can create images that look like photographs of human faces, even though the faces don't belong to any real person. These images were created by a GAN:

Images generated by a GAN created by NVIDIA.

- GANs achieve this level of realism by pairing a generator, which learns to produce the target output, with a discriminator, which learns to distinguish true data from the output of the generator.
- The generator tries to fool the discriminator, and the discriminator tries to keep from being fooled.

WHAT IS A GENERATIVE MODEL?

- "*Generative*" describes a class of statistical models that contrasts with *discriminative* models.
- Informally:
  - **Generative** models can generate new data instances.
  - **Discriminative** models discriminate between different kinds of data instances.
- More formally, given a set of data instances X and a set of labels Y:
  - **Generative** models capture the joint probability $p(X, Y)$, or just $p(X)$ if there are no labels.
  - **Discriminative** models capture the conditional probability $p(Y|X)$.
- A generative model could generate new photos of animals that look like real animals, while a discriminative model could tell a dog from a cat.
- Note that this is a very general definition. There are many kinds of generative model. GANs are just one kind of generative model.



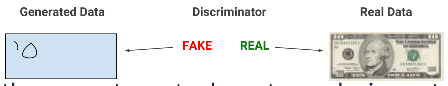
OVERVIEW OF GAN STRUCTURE

GAN STRUCTURE

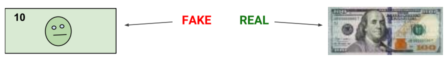
- A generative adversarial network (GAN) has two parts:
  - The **generator** learns to generate plausible data. The generated instances become negative training examples for the discriminator.
  - The **discriminator** learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results.

GAN STRUCTURE

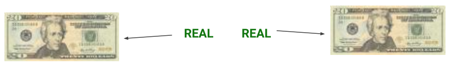
- When training begins, the generator produces obviously fake data, and the discriminator quickly learns to tell that it's fake:



- As training progresses, the generator gets closer to producing output that can fool the discriminator:

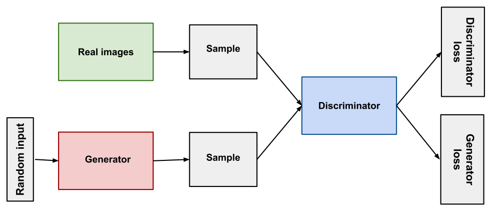


- Finally, if generator training goes well, the discriminator gets worse at telling the difference between real and fake. It starts to classify fake data as real, and its accuracy decreases.



GAN STRUCTURE

- Here's a picture of the whole system:

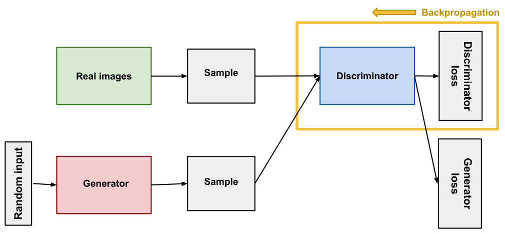


- Both the generator and the discriminator are neural networks.
- The generator output is connected directly to the discriminator input.
- Through backpropagation, the discriminator's classification provides a signal that the generator uses to update its weights.

THE DISCRIMINATOR

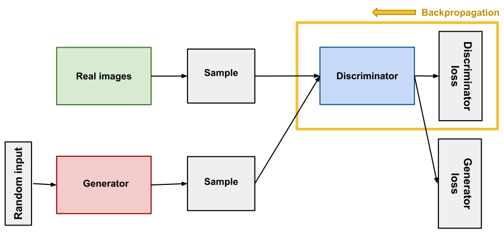
BACKPROPAGATION IN DISCRIMINATOR TRAINING

- The discriminator in a GAN is simply a classifier.
- It tries to distinguish real data from the data created by the generator.
- It could use any network architecture appropriate to the type of data it's classifying.



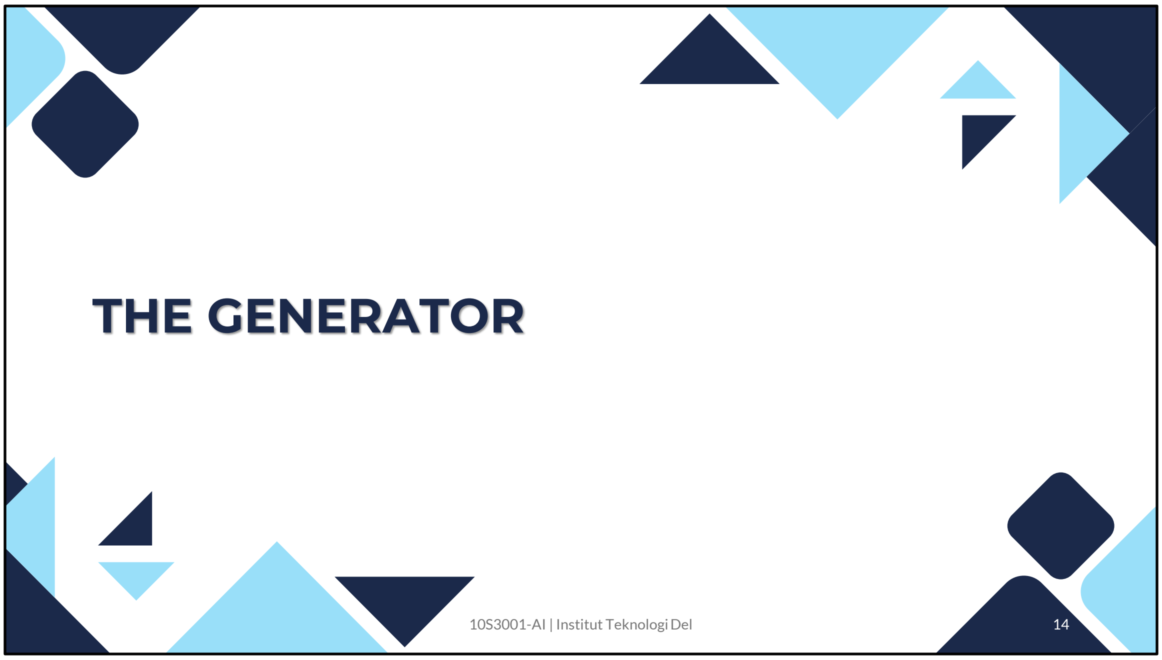
DISCRIMINATOR TRAINING DATA

- The discriminator's training data comes from two sources:
  - **Real data** instances, such as real pictures of people. The discriminator uses these instances as positive examples during training.
  - **Fake data** instances created by the generator. The discriminator uses these instances as negative examples during training.
- In the figure, the two "Sample" boxes represent these two data sources feeding into the discriminator. During discriminator training the generator does not train. Its weights remain constant while it produces examples for the discriminator to train on.



TRAINING THE DISCRIMINATOR

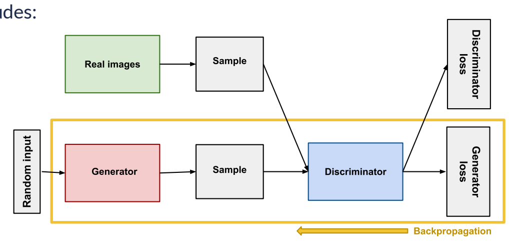
- The discriminator connects to two loss functions.
- During discriminator training, the discriminator ignores the generator loss and just uses the discriminator loss.
- We use the generator loss during generator training, as described in the next section.
- During discriminator training:
  1. The discriminator classifies both real data and fake data from the generator.
  2. The discriminator loss penalizes the discriminator for misclassifying a real instance as fake or a fake instance as real.
  3. The discriminator updates its weights through backpropagation from the discriminator loss through the discriminator network.
- In the next section we'll see why the generator loss connects to the discriminator.



THE GENERATOR

BACKPROPAGATION IN GENERATOR TRAINING

- The generator part of a GAN learns to create fake data by incorporating feedback from the discriminator.
- It learns to make the discriminator classify its output as real.
- Generator training requires tighter integration between the generator and the discriminator than discriminator training requires.
- The portion of the GAN that trains the generator includes:
  - random input
  - generator network, which transforms the random input into a data instance
  - discriminator network, which classifies the generated data
  - discriminator output
  - generator loss, which penalizes the generator for failing to fool the discriminator



INPUT

- Neural networks need some form of input.
- Normally we input data that we want to do something with, like an instance that we want to classify or make a prediction about.
- But what do we use as input for a network that outputs entirely new data instances?

RANDOM INPUT

- In its most basic form, a GAN takes random noise as its input.
- The generator then transforms this noise into a meaningful output.
- By introducing noise, we can get the GAN to produce a wide variety of data, sampling from different places in the target distribution.

DISTRIBUTION OF THE NOISE DOESN'T MATTER MUCH

- Experiments suggest that the distribution of the noise doesn't matter much, so we can choose something that's easy to sample from, like a uniform distribution.
- For convenience the space from which the noise is sampled is usually of smaller dimension than the dimensionality of the output space.
- Note: Some GANs use non-random input to shape the output. See GAN Variations.

GAN TRAINING

COMPLICATIONS

- Because a GAN contains two separately trained networks, its training algorithm must address two complications:
  - GANs must juggle two different kinds of training (generator and discriminator).
  - GAN convergence is hard to identify.

ALTERNATING TRAINING

- The generator and the discriminator have different training processes. So how do we train the GAN as a whole?
- GAN training proceeds in alternating periods:
  1. The discriminator trains for one or more epochs.
  2. The generator trains for one or more epochs.
  3. Repeat steps 1 and 2 to continue to train the generator and discriminator networks.

CONVERGENCE

- As the generator improves with training, the discriminator performance gets worse because the discriminator can't easily tell the difference between real and fake.
  - If the generator succeeds perfectly, then the discriminator has a 50% accuracy.
  - In effect, the discriminator flips a coin to make its prediction.
- This progression poses a problem for convergence of the GAN as a whole: the discriminator feedback gets less meaningful over time.
  - If the GAN continues training past the point when the discriminator is giving completely random feedback, then the generator starts to train on junk feedback, and its own quality may collapse.
- For a GAN, convergence is often a fleeting, rather than stable, state.

LOSS FUNCTIONS

LOSS FUNCTIONS

- GANs try to replicate a probability distribution.
- They should therefore use loss functions that reflect the distance between the distribution of the data generated by the GAN and the distribution of the real data.
- How do you capture the difference between two distributions in GAN loss functions?
    - This question is an area of active research, and many approaches have been proposed.
    - We'll address two common GAN loss functions here, both of which are implemented in TF-GAN:
        - minimax loss: The loss function used in the paper that introduced GANs.
        - Wasserstein loss: The default loss function for TF-GAN Estimators. First described in a 2017 paper.
- TF-GAN implements many other loss functions as well.

ONE LOSS FUNCTION OR TWO?

- A GAN can have two loss functions: one for generator training and one for discriminator training.
  - How can two loss functions work together to reflect a distance measure between probability distributions?
- In the loss schemes we'll look at here, the generator and discriminator losses derive from a single measure of distance between probability distributions.
  - In both of these schemes, however, the generator can only affect one term in the distance measure: the term that reflects the distribution of the fake data.
  - So during generator training we drop the other term, which reflects the distribution of the real data.
- The generator and discriminator losses look different in the end, even though they derive from a single formula.

MINIMAX LOSS

- In the paper that introduced GANs, the generator tries to minimize the following function while the discriminator tries to maximize it:

$$E_x[\log(D(x))] + E_z\left[\log\left(1 - D(G(z))\right)\right]$$

- In this function:
  - $D(x)$ is the discriminator's estimate of the probability that real data instance x is real.
  - E_x is the expected value over all real data instances.
  - $G(z)$ is the generator's output when given noise z.
  - $D(G(z))$ is the discriminator's estimate of the probability that a fake instance is real.
  - E_z is the expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances $G(z)$).
  - The formula derives from the cross-entropy between the real and generated distributions.
- The generator can't directly affect the $\log(D(x))$ term in the function, so, for the generator, minimizing the loss is equivalent to minimizing $\log\left(1 - D(G(z))\right)$.

MODIFIED MINIMAX LOSS

- The original GAN paper notes that the above minimax loss function can cause the GAN to get stuck in the early stages of GAN training when the discriminator's job is very easy.
- The paper therefore suggests modifying the generator loss so that the generator tries to maximize $\log D\big(G(z)\big)$.

WASSERSTEIN LOSS

- This loss function depends on a modification of the GAN scheme (called "Wasserstein GAN" or "WGAN") in which the discriminator does not actually classify instances.
  - For each instance it outputs a number.
  - This number does not have to be less than one or greater than 0, so we can't use 0.5 as a threshold to decide whether an instance is real or fake.
  - Discriminator training just tries to make the output bigger for real instances than for fake instances.
- Because it can't really discriminate between real and fake, the WGAN discriminator is actually called a "critic" instead of a "discriminator".
  - This distinction has theoretical importance, but for practical purposes we can treat it as an acknowledgement that the inputs to the loss functions don't have to be probabilities.

WASSERSTEIN LOSS

- The loss functions themselves are deceptively simple:
  - **Critic Loss**: $D(x) - D(G(z))$
    - The discriminator tries to maximize this function. In other words, it tries to maximize the difference between its output on real instances and its output on fake instances.
  - **Generator Loss**: $D(G(z))$
    - The generator tries to maximize this function. In other words, It tries to maximize the discriminator's output for its fake instances.
- In these functions:
  - $D(x)$ is the critic's output for a real instance.
  - $G(z)$ is the generator's output when given noise z.
  - $D(G(z))$ is the critic's output for a fake instance.
  - The output of critic D does not have to be between 1 and 0.
  - The formulas derive from the earth mover distance between the real and generated distributions.

COMMON PROBLEMS

COMMON PROBLEMS

- GANs have a number of common failure modes.
- All of these common problems are areas of active research.
- While none of these problems have been completely solved, we'll mention some things that people have tried.
  - Vanishing Gradients
  - Mode Collapse
  - Failure to Converge

VANISHING GRADIENTS

- Research has suggested that if your discriminator is too good, then generator training can fail due to vanishing gradients.
- In effect, an optimal discriminator doesn't provide enough information for the generator to make progress.
- Attempts to Remedy
  - **Wasserstein loss**: The Wasserstein loss is designed to prevent vanishing gradients even when you train the discriminator to optimality.
  - **Modified minimax loss**: The original GAN paper proposed a modification to minimax loss to deal with vanishing gradients.

MODE COLLAPSE (1/2)

- Usually you want your GAN to produce a wide variety of outputs.
  - For example, a different face for every random input to your face generator.
- However, if a generator produces an especially plausible output, the generator may learn to produce only that output.
  - In fact, the generator is always trying to find the one output that seems most plausible to the discriminator.
- If the generator starts producing the same output (or a small set of outputs) over and over again, the discriminator's best strategy is to learn to always reject that output.
  - But if the next generation of discriminator gets stuck in a local minimum and doesn't find the best strategy, then it's too easy for the next generator iteration to find the most plausible output for the current discriminator.
- Each iteration of generator over-optimizes for a particular discriminator, and the discriminator never manages to learn its way out of the trap.
  - As a result the generators rotate through a small set of output types. This form of GAN failure is called **mode collapse**.

MODE COLLAPSE (2/2)

- Attempts to Remedy
  - The following approaches try to force the generator to broaden its scope by preventing it from optimizing for a single fixed discriminator:
    - **Wasserstein loss**: The Wasserstein loss alleviates mode collapse by letting you train the discriminator to optimality without worrying about vanishing gradients.
      - If the discriminator doesn't get stuck in local minima, it learns to reject the outputs that the generator stabilizes on. So the generator has to try something new.
    - **Unrolled GANs**: Unrolled GANs use a generator loss function that incorporates not only the current discriminator's classifications, but also the outputs of future discriminator versions.
      - So the generator can't over-optimize for a single discriminator.

FAILURE TO CONVERGE

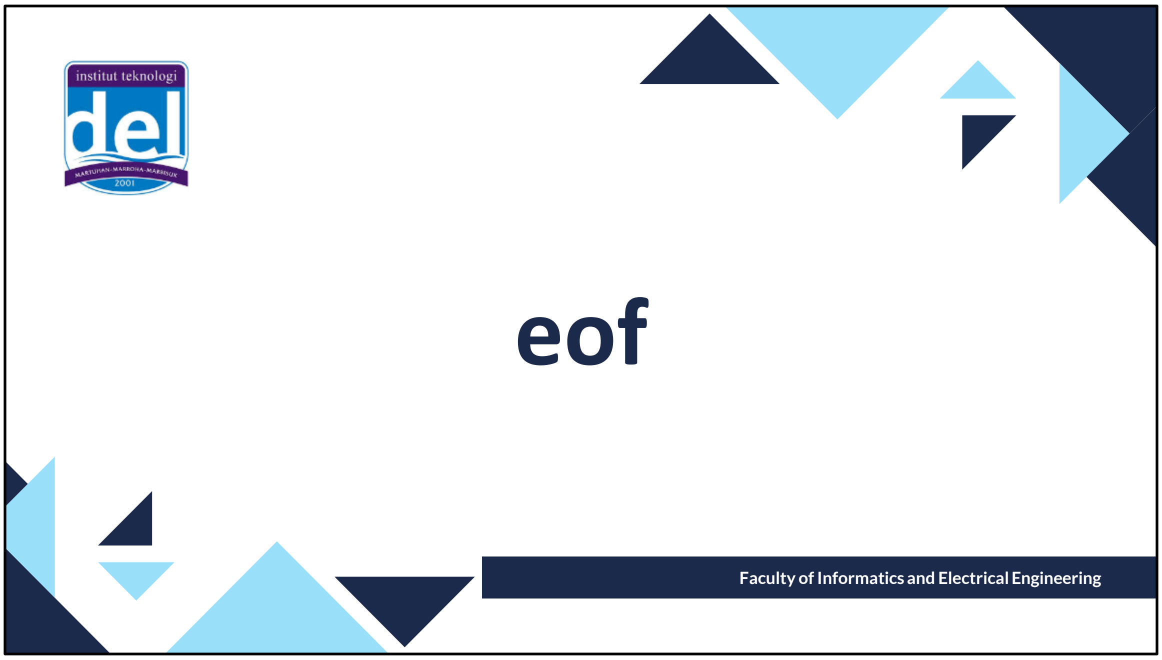
- GANs frequently fail to converge.
- Attempts to Remedy
  - Researchers have tried to use various forms of regularization to improve GAN convergence, including:
    - **Adding noise to discriminator inputs**: See, for example, Toward Principled Methods for Training Generative Adversarial Networks.
    - **Penalizing discriminator weights**: See, for example, Stabilizing Training of Generative Adversarial Networks through Regularization.

SUMMARY

- Generative models learn the data distribution to generate new data, while discriminative models learn to distinguish between classes.
- GANs can solve problems like image generation, style transfer, and data augmentation.
- The generator creates new data samples, while the discriminator evaluates their authenticity.
- Common GAN loss functions like least squares GAN and Wasserstein GAN have advantages in stability and mode collapse resistance, but may suffer from vanishing gradients or slow convergence.
- Possible solutions to common GAN training problems include using spectral normalization, feature matching, and gradient penalty.

REFERENCES

- S. J. Russell and P. Borvig, *Artificial Intelligence: A Modern Approach (4th Edition)*, Prentice Hall International, 2020.
    - Sub-Chapter 21.7. Unsupervised Learning and Treansfer Learning
- D. Foster, *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play (2nd Edition)*, O'Reilly Media, Inc., 2023.
    - Chapter 4. Generative Adversarial Networks
- GAN, https://developers.google.com/machine-learning/gan (Accessed on November 17th, 2024).



institut teknologi del

MARTUHAN-MARROHA-MARBISUK

2001

eof

Faculty of Informatics and Electrical Engineering