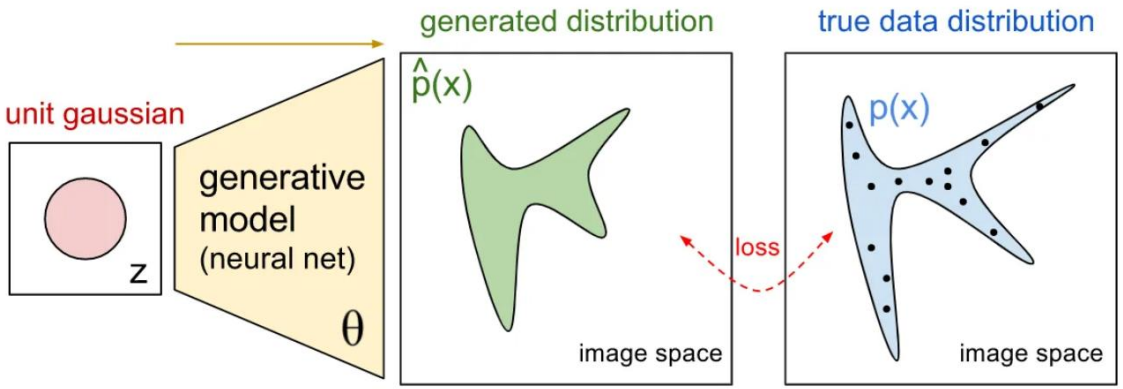
AI Lab for Wireless Communications

Week4 – Auto-Encoder

Generative Model

- Mimic the underlying distribution generating the dataset
- Approaches of generative model
  - Auto-encoder
  - Variational auto-encoder (VAE)
  - Generative adversarial network (GAN)



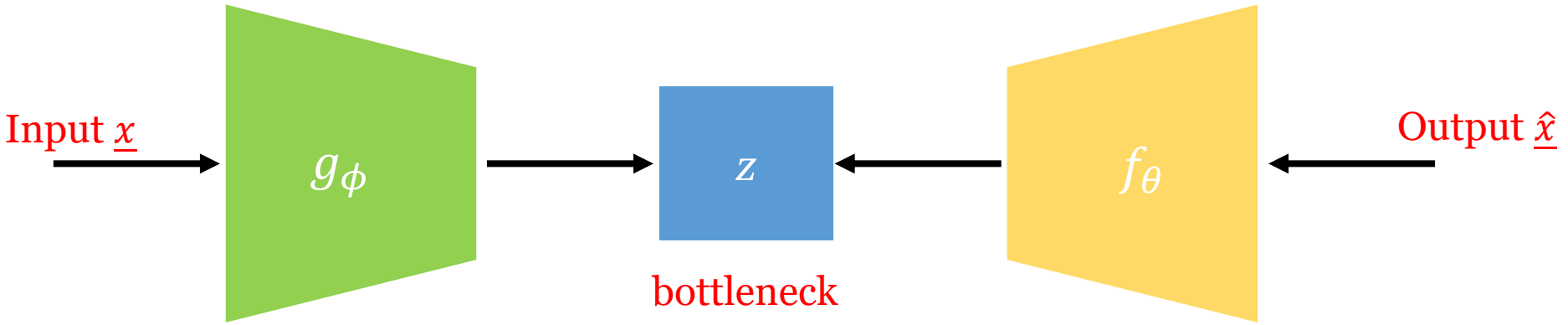
https://openai.com/research/generative-models

Auto-Encoder (1/2)

- Strictly, it is NOT a generative model
- It does not learn a distribution, but some low dimensional representation
- The loss function is to measure the similarity of $\underline{x}$ and $\underline{\hat{x}}$
- Applications
  - Dimensionality reduction
  - (Image) compression/decompression
  - (Image) denoising
  - Feature extraction
  - Clustering

Auto-Encoder (2/2)

- Find a low dimensional representative z such that we can reconstruct $\underline{x}$ from $\underline{z}$
- The parameter ϕ and θ are learned together

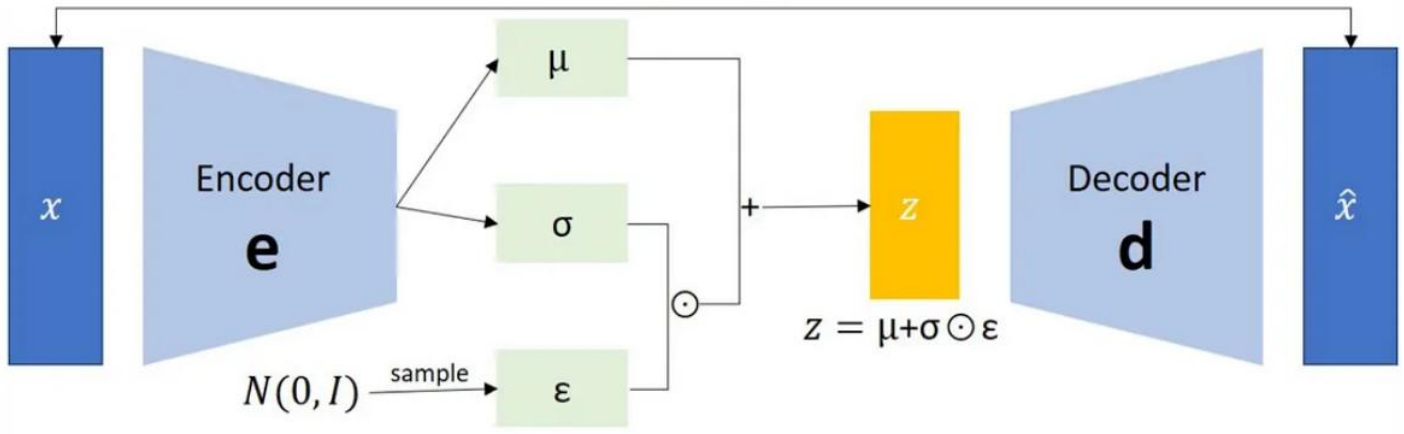


VAE (1/2)

- A generative model that learns the distribution $P_\theta(x|z)$ so that one can generate latent representation and use it to generate new $\hat{x}$
- Two goal of VAE
  - Provide low dimensional representation
  - Generate new data

VAE (2/2)

- Except the origin procedure of auto-encoder, add a parameter under normal distribution for randomness (ϵ)



https://openai.com/research/generative-models

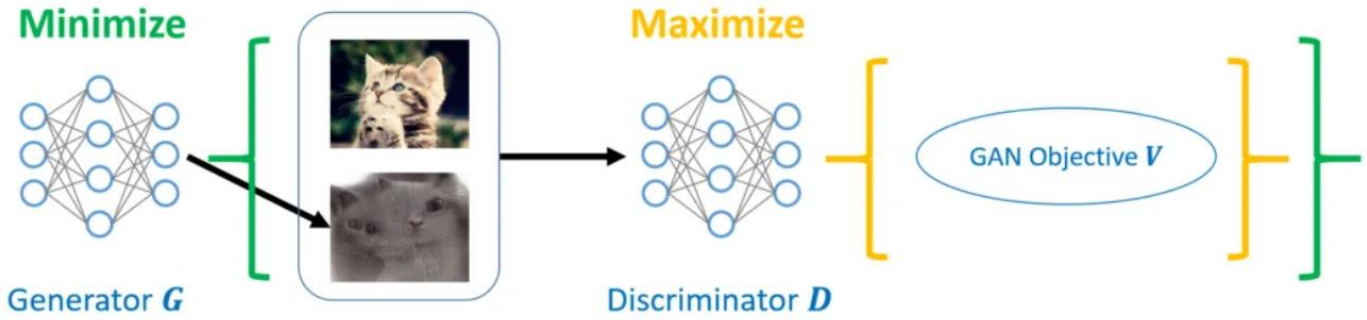
GAN (1/2)

- The goal is to learn the distribution $P_\theta(x|z)$ so that one can generate latent representation and use it to generate new $\hat{x}$

- Idea: Build another DNN model to calculate the divergence (classifying sample as either coming from the true distribution or the model's distribution

GAN (2/2)

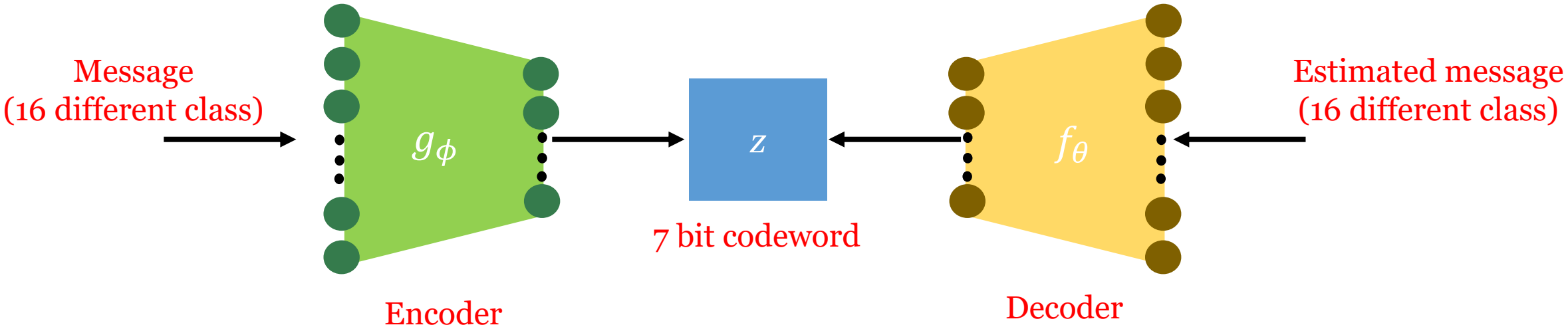
- GAN consists of two functions, generator and discriminator
- Generator generates sample to fool the discriminator (Not necessary $P_g = P_x$
- In practical, we first use DNN to train good D and use DNN to train good G. This alternating process is performing iteratively



https://openai.com/research/generative-models

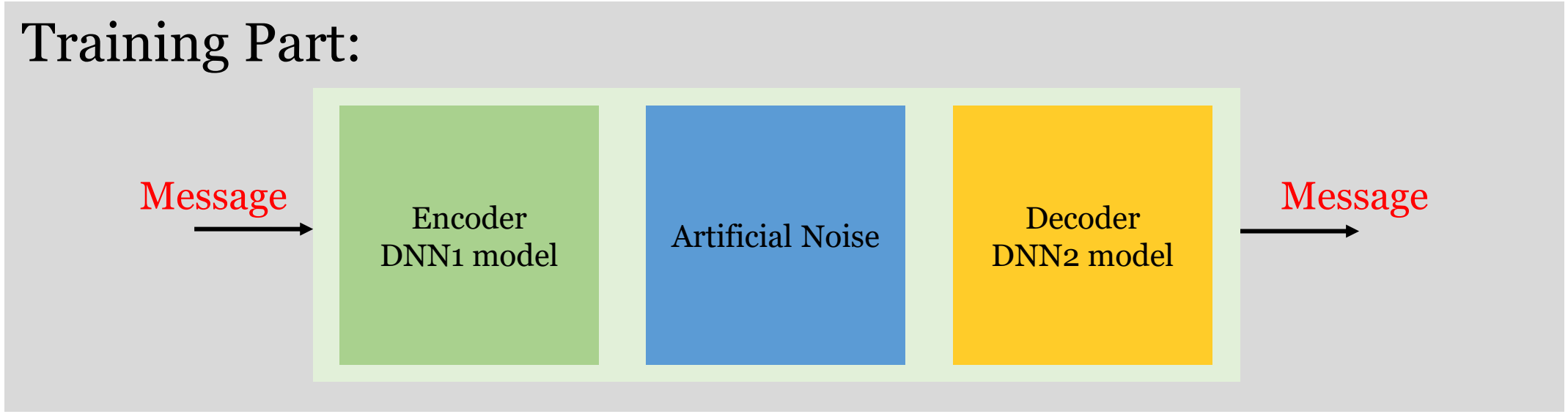
Auto-Encoder for Channel Coding

- Transform the message (class) into a low complexity representation (codeword)
- Optimize the similarity of the message (class)



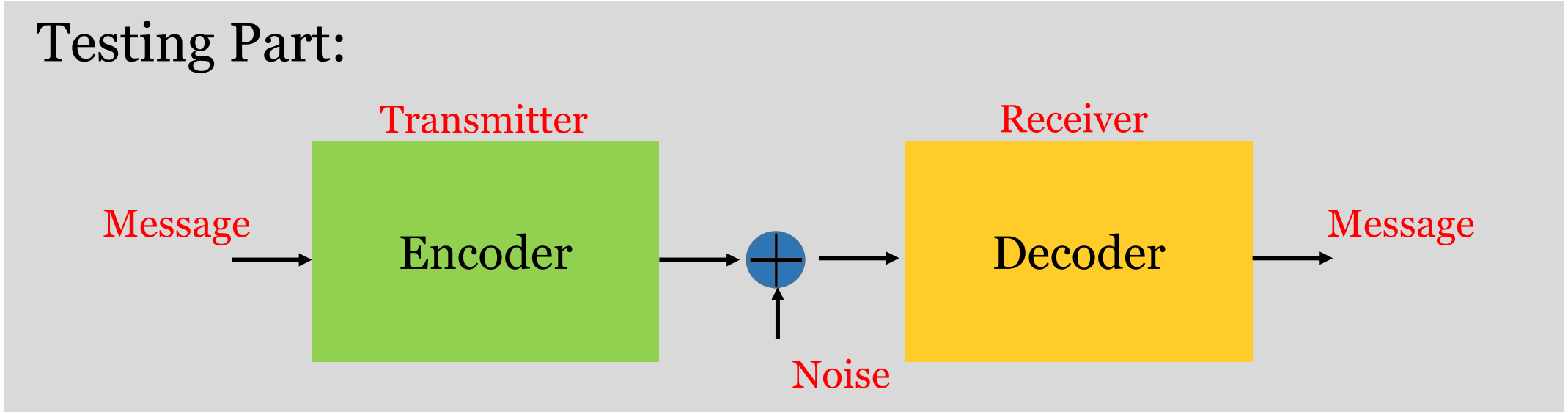
Implement Training Part

1. Build up the auto-encoder model
   (encoder (DNN1) + artificial channel noise + decoder (DNN2))
2. Train the model with training message



Implement Testing Part

1. Predict the codeword by encoding model (DNN1)
2. Add random noise to the transmitted signal
3. Predict message by the decoding model (DNN2)



Example code

- Define the structure of your model

```
# Define the autoencoder model
input_bits = keras.Input(shape=(16,))
encoded = layers.Dense(7, activation='sigmoid')(input_bits)
encoded = layers.BatchNormalization()(encoded)  # Normalize activations
noisy = layers.GaussianNoise(0.1)(encoded)  # Noise layer added
decoded = layers.Dense(16, activation='softmax')(noisy)

autoencoder = keras.Model(input_bits, decoded)
```

- Compile and train your model

```
autoencoder.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the autoencoder
autoencoder.fit(messages, messages, epochs=1000, verbose=0)
```

- Apply the trained model to do the prediction

Lab for Today

- Apply auto-encoder to channel coding problem
- Implement auto-encoder for SNR=0~6
- Demo
  - Part 1 (SNR=0~6) – Show the curve
  - Part 2 (SNR=6) – BLER competition

Top 1/3:	100
Top 2/3:	95
BLER $\leq 6 \times 10^{-3}$	90

  - Model requirement
    - Epoch ≥ 5
    - # of layers ≥ 3
      (Either Encoder or Decoder)