Rules

- Font hijau digunakan bila ada perubahan pada Assignment
- Deadline: **Jumat, 8 April 2022, at 23.55 WIB**
- Tulis identitas diri, seperti: Nama Lengkap dan NPM.
- Jangan lupa mendaftarkan kelompok anda pada tautan dibawah
  https://docs.google.com/spreadsheets/d/1G7NDzKWbGiHXDw1VBKVOytrNTOmHCYR_qpdlgRWKjaE/edit?usp=sharing
- Pastikan untuk menyertakan referensi dan/atau kolaborator bila terdapat.
- Mahasiswa diperbolehkan untuk diskusi satu sama lain. Namun, harus berhati-hati untuk tidak mencontek dan menyalin karya satu sama lain. **Semua jawaban mata kuliah harus ditulis dan dikerjakan secara mandiri oleh masing-masing mahasiswa**.
- Tulis jawaban dengan lengkap beserta cara pengerjaan atau penjelasan setiap tahap pengerjaan (untuk bagian Implementasi). Deskripsi soal bagian implementasi terdapat pada template pengerjaan soal implementasi. Perlu diingat, jangan hanya menuliskan jawaban akhir/codingan saja.
- Penalti dikenakan kepada peserta yang mengumpulkan Assignment setelah deadline berlalu dengan ketentuan:
  - Penalti 10% dari nilai total apabila mengumpulkan telat hingga Sabtu, 9 April 2022, pukul 00.55 WIB.
  - Penalti 25% dari nilai total apabila mengumpulkan telat hingga Sabtu, 9 April 2022, pukul 11.55 WIB.
  - Tidak diterima apabila lebih dari ketentuan sebelumnya.
- Kecurangan akademik berupa plagiarisme, menyontek, menyalin karya orang lain, dll akan dikenakan sanksi E dalam mata kuliah ini (hukuman maksimal). **Semua kecurangan akan dilaporkan ke fakultas, dan sanksi terakhir akan diselesaikan oleh fakultas.**
- Filenames writing format:
  Google Colab: **AS_Group Number.ipynb**

1 Artificial Neural Networks

Pada bagian ini akan melakukan klasifikasi citra menggunaan ANN, untuk dataset, model, dan hyperparameter yang digunakan dapat dilihat dibawah.

Dataset:

- **Dataset untuk group ganjil**: https://www.tensorflow.org/datasets/catalog/horses_or_humans
- **Dataset untuk group genap**: https://www.tensorflow.org/datasets/catalog/rock_paper_scissors

Hyperparamenter:

- Optimizer: Adam (learning rate=0.001)
- Loss: Cross Entropy
- Activation Function: ReLu
- Batch Size: 32
- Epoch: 50

Model:

	Hidden Layer	**Neuron**
Model 1	3	[32, 64, 128]
Model 2	4	[16, 32, 64, 128]
Model 3	5	[8, 16, 32, 64, 128]

(a) Split data menjadi train, validation, dan test set.

(b) Train setiap model seperti ketentuan yang sudah tertera. Lakukan juga proses testing.

(c) Evaluasi model menggunakan akurasi, sensitivity, specificity, dan F1 score.

(d) Diskusikan hasil dari setiap model.

2 Convolutional Neural Networks

Arsitektur Model:

- Convolutional Layer 2D (16 maps, kernel 3x3, padding=same/1)
- BatchNormalization 2D
- Activation Function (ReLu)
- Maxpool 2D (2x2)
- Convolutional Layer 2D (32 maps, kernel 3x3, padding=same/1)

- BatchNormalization 2D
- Activation Function (ReLu)
- Maxpool 2D (2x2)
- Convolutional Layer 2D (64 maps, kernel 3x3, padding=same/1)
- BatchNormalization 2D
- Activation Function (ReLu)
- Maxpool 2D (2x2)
- Convolutional Layer 2D (128 maps, kernel 3x3, padding=same/1)
- BatchNormalization 2D
- Activation Function (ReLu)
- Maxpool 2D (2x2)
- Flatten
- Hidden layer (1000 neurons)
- Activation Function (ReLu)
- Output layer (sesusai jumlah kelas)
- Activation Function (Softmax)

(a) Gunakan dataset yang sama dengan bagian sebelumnya. Split data menjadi train, validation, dan test set.

(b) Lakukan hyperparameter tunning mulai dari batch size=[8, 32, 64], learning rate=[0.0001, 0.001, 0.01], dan epoch=[10, 30, 50].

(c) Evaluasi model menggunakan akurasi, sensitivity, specificity, dan F1 score.

(d) Coba gunakan pre-trained model ResNet50 (docs: https://www.tensorflow.org/api_docs/python/tf/k

(e) Gunakan hyperpaameter yang optimal dari point b, hanya saja epoch yang digunakan hanya 10.

(f) Evaluasi model menggunakan akurasi, sensitivity, specificity, dan F1 score.

(g) Diskusikan hasil dari model yang digunakan dan model pre-trained.

3 Variational Autoencoder

Pada bagian ini, anda diminta untuk membangun variational autoencoder (VAE) dari awal. Untuk keperluan tugas ini, anda harus menggunakan dataset KMNIST untuk melatih VAE anda.

- PyTorch KMNIST: https://pytorch.org/vision/main/generated/torchvision.datasets.KMNIST.html
- Tensorflow KMNIST: https://www.tensorflow.org/datasets/catalog/kmnist

Gunakan 60000 data untuk keperluan training dan 10000 sisanya untuk keperluan evaluasi.

VAE memandang encoder dan decoder sebagai suatu model probabilistik teramortisasi. Encoder merepresentasikan $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, dengan $\boldsymbol{\phi}$ menyatakan weight encoder. Selanjutnya, decoder merepresentasikan $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, dengan $\boldsymbol{\theta}$ menyatakan weight decoder. Selain itu, terdapat distribusi prior $p(\mathbf{z})$ yang berperan meregularisasi $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ seketat mungkin terhadap distribusi prior. Kondisi tersebut direalisasikan melalui $\mathrm{KL}\left[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right]$. Fungsi objektif VAE direalisasikan dengan meminimalkan objektif berikut:

$$\mathcal{L} = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[-\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] + \mathrm{KL}\left[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right]$$

KL menyatakan Kullback-Leibler divergence. Suku pertama merupakan likelihood yang bertujuan menemukan parameter VAE yang sesuai terhadap data yang diobservasi. Suku kedua memaksa $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ semirip mungkin terhadap $p(\mathbf{z})$ secara statistik. Penjelasan lebih lanjut mengenai VAE dapat merujuk https://arxiv.org/pdf/1312.6114.pdf

Arsitektur dan Hyperparameter

Encoder	Decoder
Conv2D (in=1, out=8, kernel=3, stride=2, pad=1) ReLU	Linear (in=dim, out=128) ReLU
Conv2D (in=8, out=16, kernel=3, stride=2, pad=1) BatchNorm2D ReLU	Linear (in=128, out=32*3*3=288) ReLU
Conv2D (in=16, out=32, kernel=3, stride=2, pad=0) ReLU	ConvTranspose2D (in=32, out=16, kernel=3, stride=2, pad=0) BatchNorm2D ReLU
Linear (in=32*3*3=288, out=128) ReLU	ConvTranspose2D (in=16, out=8, kernel=3, stride=2, pad=1) BatchNorm2D ReLU
Linear (in=128, out=dim)	ConvTranspose2D (in=8, out=1, kernel=3, stride=2, pad=1)

Sebagai pengingat, luaran encoder berupa parameter distribusi $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. Pada tugas ini, diasumsikan bahwa $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ berupa distribusi multivariate Gaussian. Karena itu, lapisan terakhir encoder berupa mean $\boldsymbol{\mu}$ (`linear[:128]`) dan matriks covariance $\boldsymbol{\Sigma}$ (`linear[128:]`). Selanjutnya, diasumsikan bahwa luaran decoder berupa distribusi multivariat Gaussian dengan covariance berupa isotropic Gaussian (Lihat asumsi pada akhir bagian ini).

Untuk memudahkan implementasi, silahkan gunakan hyperparameter berikut:

Hyperparameter	value
Epoch	50 atau lebih
batch size	128 atau lebih
learning rate	1e-3
optimizer	Adam (Adaptive momentum)
seed generator	42
dimensionalitas latent variable	128

Seed generator memastikan luaran model anda statis untuk setiap iterasi sehingga memudahkan anda untuk mereproduksi VAE yang anda bangun.

Training

VAE dapat dilatih dengan bantuan reparameterization trick. Teknik sampling ini melibatkan noise $\boldsymbol{\epsilon}$ yang disampel dari distribusi Gaussian standar $\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, c\mathbf{I})$ dengan c merupakan variance.

Algorithm 1 Training Algorithm for Variational Autoencoder (1 iterasi/1 epoch)

Require: Encoder E, Decoder D, c
  for each training iteration **do**
    for each batch **do**
      - Sample $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$ from $p(\mathbf{x})$
      - $\boldsymbol{\mu}_{\mathbf{z}}^{(i)}, \boldsymbol{\Sigma}_{\mathbf{z}}^{(i)} = E_{\boldsymbol{\phi}}(\mathbf{x}^{(i)})$ (compute $q_{\boldsymbol{\phi}}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})$)
      - Sample $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$
      - $\mathbf{z}^{(i)} = \boldsymbol{\mu}_{\mathbf{z}}^{(i)} + \boldsymbol{\epsilon}^{(i)}\ \mathbf{L}_{\mathbf{z}}^{(i)}$ (reparameterization trick)
      - $\boldsymbol{\mu}_{\hat{\mathbf{x}}}^{(i)} = D_{\boldsymbol{\theta}}(\mathbf{z}^{(i)})$ (compute $p_{\boldsymbol{\theta}}(\hat{\mathbf{x}}^{(i)}|\mathbf{z}^{(i)})$)
      - $\hat{\mathcal{L}} = \frac{1}{N}\sum_{i=1}^{N}\left[-\log\mathcal{N}\left(\hat{\mathbf{x}}^{(i)} = \mathbf{x}^{(i)}; \boldsymbol{\mu}_{\hat{\mathbf{x}}}^{(i)}, c\mathbf{I}\right) + \mathrm{KL}\left[\mathcal{N}(\mathbf{z}^{(i)}; \boldsymbol{\mu}_{\mathbf{z}}^{(i)}, \boldsymbol{\Sigma}_{\mathbf{z}}^{(i)}) \| \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})\right]\right]$
      - Minimize $\hat{\mathcal{L}}$ w.r.t. $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ (stochastic Adam)
    end for

  end for

Assumsi:

- $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ merupakan distribusi multivariate Gaussian terfaktorisasi, yaitu $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \prod_j q_{\phi_j}(\mathbf{z}_j|\mathbf{x}) = \prod_j \mathcal{N}(\mu_j, \sigma_j^2)$.
- $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$
- $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{\hat{\mathbf{x}}}, c\mathbf{I})$ with $c = 0.0001$.
- $\boldsymbol{\Sigma}_{\mathbf{z}} = \mathbf{L}_{\mathbf{z}}^{\mathrm{T}}\mathbf{L}_{\mathbf{z}}$ (standard deviasi)

Tips:

- Berdasarkan asumsi, log-likelihood $-\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ dapat disimplifikasi menjadi Euclidan distance.
- Ketika melakukan reparameterization trick, sebaiknya luaran encoder merepresentasikan log covariance demi menjaga stabilitas numerik.

Evaluasi

Lakukan image reconstruction terhadap 10000 data yang sudah anda alokasikan sebelumnya. Untuk setiap $\mathbf{x}_{test}$, kita ingin mendapatkan $\hat{\mathbf{x}}_{test}$ melalui VAE. Anda tidak perlu menampilkan seluruh hasil rekonstruksi. Selanjutnya, bandingkan $\mathbf{x}_{test}$ and $\hat{\mathbf{x}}_{test}$ menggunakan **MSE** and **cosine-similarity**.

$$\text{MSE}(\hat{\mathbf{x}}_{test}, \mathbf{x}_{test}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{x}}_{test}^{(i)} - \mathbf{x}_{test}^{(i)})^2$$

$$\cos(\hat{\mathbf{x}}_{test}, \mathbf{x}_{test}) = \frac{1}{N}\sum_{i=1}^{N}\frac{\hat{\mathbf{x}}_{test}^{(i)} \cdot \mathbf{x}_{test}^{(i)}}{\|\hat{\mathbf{x}}_{test}^{(i)}\|\|\mathbf{x}_{test}^{(i)}\|}$$

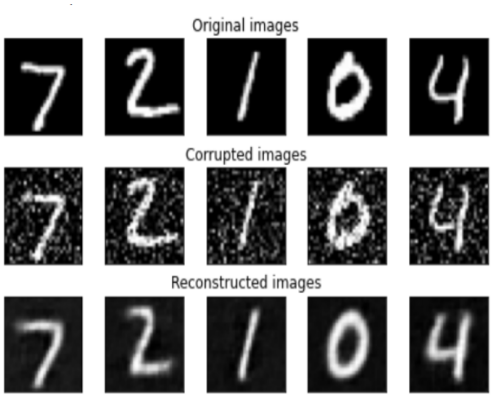

Figure 1: Contoh image reconstruction pada dataset MNIST.

- $\boldsymbol{\mu}_{\mathbf{z}_{test}}^{(i)}, \boldsymbol{\Sigma}_{\mathbf{z}_{test}}^{(i)} = E_{\boldsymbol{\phi}}(\hat{\mathbf{x}}^{(i)})$
- $\mathbf{z}_{test} = \{\boldsymbol{\mu}_{\mathbf{z}_{test}}^{(i)} + \boldsymbol{\epsilon}^{(i)} \cdot \boldsymbol{\Sigma}_{\mathbf{z}_{test}}^{(i)}\}_{i=1}^{N=10000}$
- $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})$
- $\hat{\mathbf{x}}_{test} \sim D_{\boldsymbol{\theta}}(\mathbf{z}_{test})$

4 Wassertein GAN

Pada bagian ini anda diminta untuk mengimplementasikan varian lain dari GAN, yang dibangun berdasarkan teori optimal transport. Untuk keperluan tugas ini, anda akan menggunakan dataset Fashion-MNIST:

- Tensorflow : https://www.tensorflow.org/datasets/catalog/fashion_mnist
- PyTorch: https://pytorch.org/vision/main/generated/torchvision.datasets.FashionMNIST.html

Cukup gunakan data training untuk melatih WGAN anda.

Wasserstein GAN menggunakan metrik Wasserstein distance untuk menggantikan optimisasi min-max yang secara teoritis merupakan Jensen-Shannon divergence. Pada implementasinya, metrik ini diaproksimasi melalui Kantorovich-Rubinstein duality:

$$W(P_r, P_{\boldsymbol{\theta}}) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim P_r}[f(\mathbf{x})] - \mathbb{E}_{x \sim \boldsymbol{\theta}}[f(\mathbf{x})]$$

Berdasarkan persamaan tersebut, supremum dipilih dari fungsi 1-Lipschitz. Selanjutnya, f menyatakan fungsi kritik yang juga merupakan neural network. Pada model GAN, f merepresentasikan diskriminator. Sementar pada kasus WGAN, luaran f bukan probabilitas sehingga penamaan diskriminator diubah. Misalkan $\boldsymbol{\omega}$ menyatakan parameter weight f dan $g_{\boldsymbol{\theta}}$ menyatkan generator yang diparameterisasi oleh $\boldsymbol{\theta}$. Dengan asumsi bahwa terdapat $\boldsymbol{w} \in \mathcal{W}$ yang merupakan supremum dari fungsi Lipschitz, objektif WGAN dapat dituliskan sebagai berikut:

$$\nabla_{\boldsymbol{\theta}}\, W(P_r, P_{\boldsymbol{\theta}}) = \nabla_{\boldsymbol{\theta}} \left(\mathbb{E}_{\mathbf{x} \sim P_r}[f_{\boldsymbol{\omega}}(\mathbf{x})] - \mathbb{E}_{z \sim P_Z}[f_{\boldsymbol{\omega}}(g_{\boldsymbol{\theta}})(\mathbf{z})]\right)$$

Variabel $\mathbf{z}$ menyatakan vektor noise yang disampel dari distribusi prior. Pada tugas ini $\mathbf{z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. Penjelasan lebih lengkap mengenai WGAN dapat anda temukan di https://arxiv.org/pdf/1701.07875.pdf.

Arsitektur dan hyperparameter

Generator	**Fungsi kritik**
Linear (in=noise-dim (128), out=256) Leaky ReLU (0.2)	Linear (in=dim, out=1024) Leaky ReLU (0.2) Dropout (0.3)
Linear (in=256, out=512) Leaky ReLU (0.2)	Linear (in=1024, out=512) Leaky ReLU(0.2) Dropout(0.3)
Linear (in=512, out=1024) Leaky ReLU (0.2)	Linear in=512, out=256 Leaky ReLU(0.2) Dropout(0.3)
Linear (in=1024, out=input-dim) Tanh	Linear (input=256, dim=1)

Parameter dropout menyatakan probabilitas neuron aktif terlibat pada proses pemelajaran. Berikut hyperparameter yang anda harus gunakan untuk mengimplementasikan WGAN anda:

Hyperparameter	**value**
Epoch	50 atau lebih
batch size	128 atau lebih
learning rate	5e-5
optimizer	RMSProp
seed generator	42
dimensionalitas latent variable	128
iterasi fungsi kritik (n_critic)	5

Training

Algorithm 2 Training Algorithm for Wasserstein GAN

Require: Generator g, critic-function f, clipping parameter c, learning rate α, number of critic n_{critic}, batch size m

for each training iteration **do**

for $t = 0, \cdots, n_{critic}$ **do**

- Sample $\{\mathbf{x}^{(i)})\}_{i=1}^{N}$ from P_r
- Sample $\{\mathbf{z}^{(i)})\}_{i=1}^{N}$ from P_Z
- $\hat{\mathcal{L}}_{\boldsymbol{\omega}} \leftarrow \nabla_{\boldsymbol{\omega}} \left[\frac{1}{m}\sum_{i=1}^{m} f_{\boldsymbol{\omega}}(\mathbf{x}^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_{\boldsymbol{\omega}}(g_{\boldsymbol{\theta}}(\mathbf{z}^{(i)}))\right]$
- $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \alpha\,.\,\text{RMSProp}(\hat{\mathcal{L}}_{\boldsymbol{\omega}}, \boldsymbol{\omega})$
- Clip $(\boldsymbol{\omega}, -c, c)$

end for

- Sample $\{\mathbf{z}^{(i)}\}_{i=1}^{m}$ from P_Z
- $\hat{\mathcal{L}}_{\boldsymbol{\theta}} \leftarrow -\nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} f_{\boldsymbol{\omega}}(g_{\boldsymbol{\theta}}(\mathbf{z}^{(i)}))$
- $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha.\text{RMSProp}(\hat{\mathcal{L}}_{\theta}, \boldsymbol{\theta})$

end for

Evaluasi

Evaluasi dilakukan secara kualitatif. Anda diminta untuk membangkitkan 100 sampel melalui generator WGAN yang sudah anda latih sebelumnya. Terlebih dahulu anda perlu mendapatkan 100 sampel noise $\mathbf{z}$. Selanjutnya, anda perlu mentransformasi noise tersebut melalui generator. Anda dapat menampilkan 100 citra yang sudah anda bangkitkan dalam format grid berukuran 10×10. Selama proses evaluasi, anda tidak memerlukan fungsi kritik f.

References

[1] Arjovsky, M., Chintala, S., & Bottou, L. (2017, July). Wasserstein generative adversarial networks. In International conference on machine learning (pp. 214-223). PMLR.

[2] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.