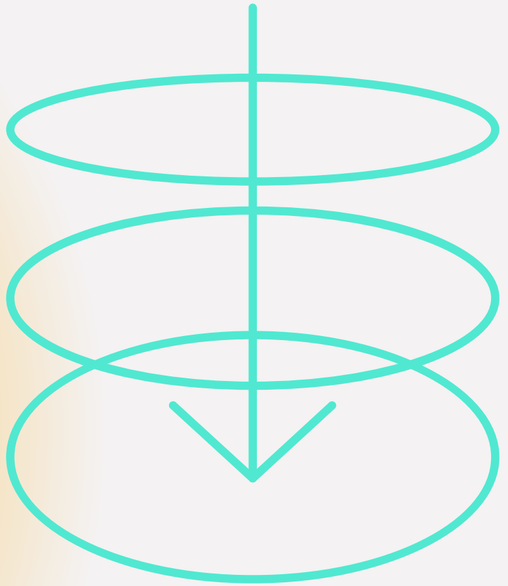
Final Project



Anggara Saputra 5025211241

Ketut Arda Putra Mahotama 5025211235

00 - Deep Learning

Deep learning merupakan subbidang machine learning yang algoritmanya terinspirasi dari struktur otak manusia. Struktur tersebut dinamakan Artificial Neural Networks atau disingkat ANN. Pada dasarnya, ia merupakan jaringan saraf yang memiliki tiga atau lebih lapisan ANN. Ia mampu belajar dan beradaptasi terhadap sejumlah besar data serta menyelesaikan berbagai permasalahan yang sulit diselesaikan dengan algoritma machine learning lainnya.

Manfaat :

- Dapat memproses unstructured data seperti teks dan gambar.
- Dapat mengotomatisasi proses ekstraksi fitur tanpa perlu melakukan proses pelabelan secara manual.
- Memberikan hasil akhir yang berkualitas.
- Dapat mengurangi biaya operasional.
- Dapat melakukan manipulasi data dengan lebih efektif.

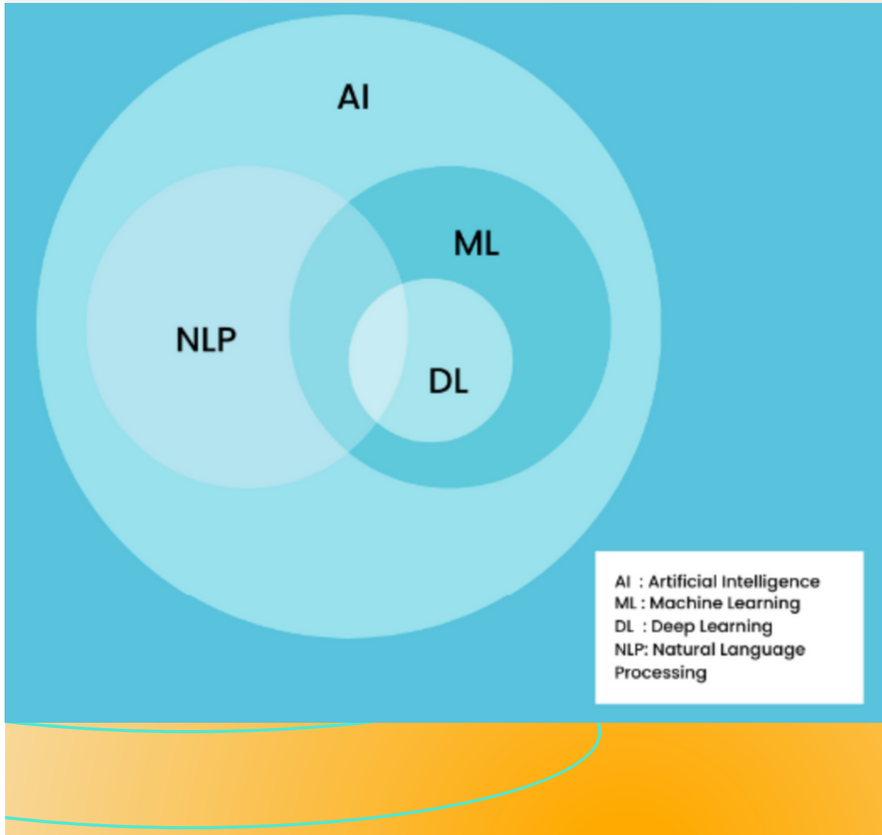
Penerapan :

- Pengenalan Gambar
- Pengenalan Suara
- Memberikan hasil akhir yang berkualitas.
- Natural language processing
- Deteksi Anomali

01 - What is NLP?

Natural Language Processing atau disingkat NLP adalah bagian dari Artificial Intelligence (AI) yang berkaitan dengan memberikan kemampuan pada komputer untuk memahami bahasa alami manusia. Seperti tulisan maupun suara yang sering digunakan oleh manusia dalam percakapan sehari-hari.

NLP diciptakan dengan menggabungkan linguistik komputasi bersama model statistik, machine learning, dan deep learning. Semua unsur tersebut kemudian digunakan untuk membantu komputer memproses (processing) data teks maupun audio manusia. Membuatnya memahami data tersebut hingga mendapatkan makna penuh, lengkap dengan maksud dan sentimennya.



02 - NLP Types

Rules-based System

Jenis algoritma pertama yang digunakan dalam NLP adalah menggunakan aturan linguistik yang dirancang dengan teliti. Rules-based system telah digunakan pada awal pengembangan NLP dan masih digunakan hingga saat ini.

Machine Learning-based System

Jenis algoritma kedua menggunakan metode statistik. Di mana komputer akan belajar melakukan tugas berdasarkan data pelatihan yang diberikan, dan menyesuaikan algoritma mereka saat lebih banyak data yang diproses. Machine learning-based system menggunakan kombinasi machine learning, deep learning, serta neural networks. Algoritma NLP jenis ini akan mengasah aturan mereka sendiri melalui pemrosesan dan pembelajaran berulang.

Klasifikasi Genre Berdasarkan Sinopsis

How it's work

03.1 - Pengumpulan dan Praproses Data

Kumpulkan sejumlah data teks yang berisi deskripsi atau sinopsis film beserta dengan label genre filmnya. Praproses data dengan langkah-langkah seperti tokenisasi, normalisasi, penghapusan stopwords, dan pembersihan teks dari karakter atau simbol yang tidak penting.

How it's work

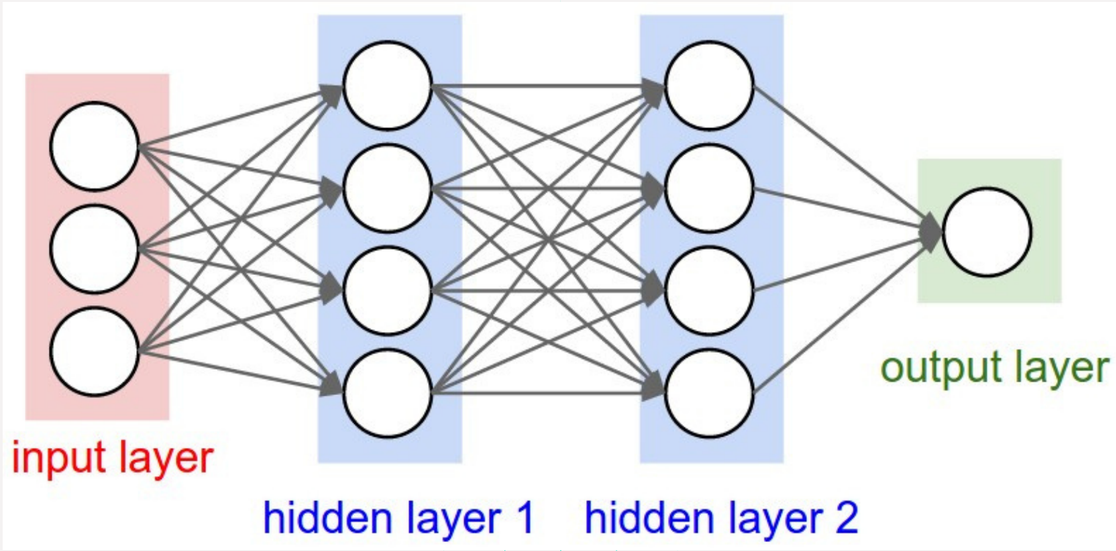
03.2 - Representasi Teks

Ubah teks menjadi representasi numerik yang dapat diproses oleh model deep learning. Salah satu pendekatan yang umum adalah menggunakan metode Word Embedding seperti Word2Vec atau fastText untuk mengubah setiap kata dalam teks menjadi vektor numerik. Setelah itu melaui model inferSent akan menghasilkan Sentence Embedding yang dapat diproses oleh model klasifikasi

How it's work

03.3 - Pembangunan Model

Gunakan arsitektur deep learning Fully Connected Neural Network untuk membangun model klasifikasi teks. Model ini akan mengambil representasi numerik dari teks dan mempelajari pola-pola yang terkait dengan genre film.



How it's work

03.4 - Pelatihan Model

Latih model dengan menggunakan data latihan yang terdiri dari teks dan label genre film. Model akan memperbarui bobot-bobotnya berdasarkan perbedaan antara prediksi model dan label yang benar. Proses pelatihan menggunakan metode backpropagation dan optimisasi untuk meningkatkan kinerja model.

How it's work

03.5 - Evaluasi dan Penyetelan Model

Evaluasi kinerja model menggunakan data uji yang terpisah untuk mengukur akurasi dan kinerja klasifikasi genre film. Jika kinerja model belum memuaskan, dapat dilakukan penyetelan parameter model atau perubahan arsitektur untuk meningkatkan hasilnya.

How it's work

03.6 - Penggunaan Model

Setelah model dilatih dan dievaluasi, model dapat digunakan untuk mengidentifikasi genre film dari teks baru. Teks baru akan diolah melalui langkah-langkah praproses yang sama dan kemudian diberikan ke model untuk melakukan klasifikasi genre film.

04 - Conclusions

Dengan menggunakan metode deep learning dan NLP seperti yang dijelaskan di atas, model dapat mempelajari pola-pola yang terkait dengan genre film dari data latihan, dan kemudian dapat mengidentifikasi genre film dari teks baru berdasarkan pola-pola tersebut.

Proses Pembersihan dan Klafisikasi Data

- Membaca dataset dan memasukan ke dataframe pandas

```
# Membaca file .csv
df = pd.read_csv('./imdb_top_1000.csv')
```

- Filter berdasarkan panjang sinopsis dan menghapus karakter-karakter unicode pada sinopsis

```
# Mengambil hanya sinopsis (Overview) dan genre
df = df[['Overview', 'Genre']]
# Memisah string genre dengan delimiter "," menjadi array of string
df['Genre'] = df['Genre'].apply(lambda x : x.split(', '))
# Filter berdasarkan panjang sinopsis
df = df[df['Overview'].apply(lambda x : len(x)) > 100]
# Menghapus karakter unicode pada sinopsis
df['Overview'] = df['Overview'].str.replace('[^\x00-\x7F]','', regex=True)
```

Proses Pembersihan dan Klafisikasi Data

- Melakukan reset index

```
df = df.reset_index()[['Overview', 'Genre']]
df.shape[0]
```

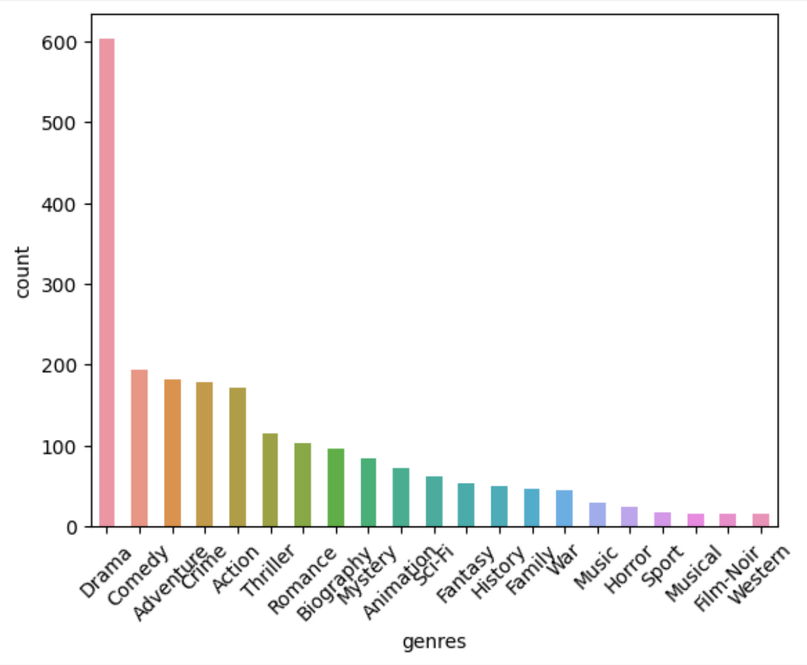
- # Definisi fungsi untuk plotting genre

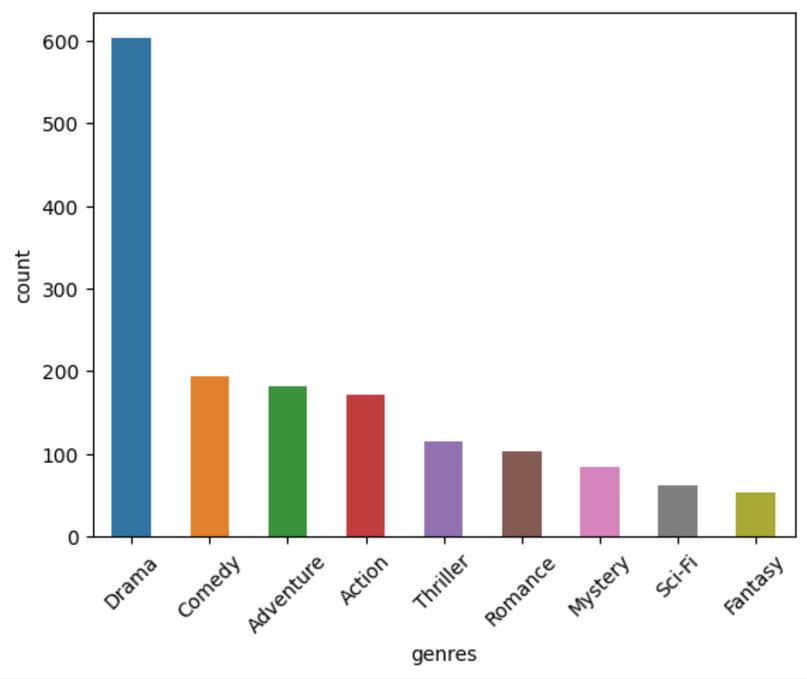
```
def plot_genres(genres_data):
 genres = np.unique(np.concatenate(genres_data['Genre']))
 genre_count = {genre: np.count_nonzero(np.concatenate(genres_data['Genre']) == genre) for genre in genres}

 genre_count = {
 "genres": list(genre_count.keys()),
 "count": list(genre_count.values())
 }

 genre_count = pd.DataFrame(genre_count)
 genre_count = genre_count.sort_values(by="count", ascending=False)

 genre_plot = sb.barplot(genre_count, x="genres", y="count", width=0.5)
 genre_plot.tick_params(axis="x", rotation=45)
plot_genres(df)
```





List genre yang paling relevan dan Filter array genre sesuai genre yang relevan

```
GENRES = ['Drama', 'Comedy', 'Adventure', 'Action',
'Thriller', 'Romance', 'Mystery', 'Sci-Fi', 'Fantasy']
```

```
def filter_genres(input_genre, genre_list):
 return [genre for genre in input_genre if genre in genre_list]

df['Genre'] = df['Genre'].apply(lambda x: filter_genres(x, GENRES))
plot_genres(df)
```

	Overview	Genre
0	Two imprisoned men bond over a number of years...	[Drama]
1	An organized crime dynasty's aging patriarch t...	[Drama]
2	When the menace known as the Joker wreaks havo...	[Action, Drama]
3	The early life and career of Vito Corleone in ...	[Drama]
4	A jury holdout attempts to prevent a miscarria...	[Drama]
...	...	...
842	Over two "typical" days in the life of The Bea...	[Comedy]
843	A young New York socialite becomes interested ...	[Comedy, Drama, Romance]
844	In Hawaii in 1941, a private is cruelly punish...	[Drama, Romance]
845	Several survivors of a torpedoed merchant ship...	[Drama]
846	A man in London tries to help a counter-espion...	[Mystery, Thriller]

Model

- Menginstansiasi model infersent untuk menghasilkan sentence embeddings(Vocab size = 5000)

```
from models import InferSent
MODEL_PATH = "encoder/infersent2.pkl"
params_model = {'bsize': 64, 'word_emb_dim': 300, 'enc_lstm_dim':
2048,
 'pool_type': 'max', 'dpout_model': 0.0, 'version': 2}
model = InferSent(params_model)
model.load_state_dict(torch.load(MODEL_PATH))

W2V_PATH = 'fastText/crawl-300d-2M.vec'
model.set_w2v_path(W2V_PATH)

model.build_vocab_k_words(K=50000)
```

- Konversi genre pada dataframe menjadi array of boolean

```
# Membaca dataset yang telah dibersihkan
dataset = pd.read_json('clean.json')

# List genre yang paling relevan
GENRES = ['Drama', 'Comedy', 'Adventure', 'Action', 'Thriller', 'Romance', 'Mystery',
'Sci-Fi', 'Fantasy']

# Konversi menjadi array of boolean agar dapat diproses pytorch
dataset['Genre'] = dataset['Genre'].apply(lambda x: np.isin(GENRES, x))
```

Model

- # Menghasilkan sentence embedding

```
embeddings = model.encode(dataset['Overview'].to_numpy(), bsize=128,
tokenize=False, verbose=True)
==========================================================================
```

Nb words kept : 21420/24424 (87.7%)
c:\Users\ardap\OneDrive\Desktop\Koding\Kuliah\Semester 4\Keluarga Berencana\Kelompok\FP-KB\models.py:207: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
** sentences = np.array(sentences)[idx_sort]**
Speed : 57.9 sentences/s (cpu mode, bsize=128)

- Hasil konversi menjadi array of boolean

```
dataset['Embeddings'] = embeddings.tolist()
dataset = dataset[['Embeddings', 'Genre']]
dataset
==========================================================================
```

	Embeddings	Genre
0	[0.007468892261385918, 0.3183546066284179 7, 0....	[True, False, False, False, False, False, Fals...
1	[0.007468892261385918, 0.0813099965453147 9, 0....	[True, False, False, False, False, False, Fals...
2	[0.007468892261385918, 0.0382896773517131 8, 0....	[True, False, False, True, False, False, False...
3	[0.007468892261385918, 0.0310220345854759 2, 0....	[True, False, False, False, False, False, Fals...
4	[0.007468892261385918, 0.0258408002555370 33, 0...	[True, False, False, False, False, False, Fals...
...	...	...
842	[0.007468892261385918, 0.1860463917255401 6, 0....	[False, True, False, False, False, False, Fals...
843	[0.007468892261385918, 0.1551827788352966 3, 0....	[True, True, False, False, False, True, False,...
844	[0.007468892261385918, 0.0699759721755981 4, 0....	[True, False, False, False, False, True, False...
845	[0.007468892261385918, 0.0235678460448980 33, 0...	[True, False, False, False, False, False, Fals...
846	[0.007468892261385918, 0.1772287487983703 6, 0....	[False, False, False, False, True, False, True...

847 rows × 2 columns

Model

- Memisahkan data untuk training dan testing dengan split 10% (90% training, 10% testing)

```
# Mengasilkan dataset untuk training dan testing (validasi)
train_ds, test_ds = train_test_split(dataset, test_size=0.1)
train_ds = train_ds.reset_index(drop=True)
test_ds = test_ds.reset_index(drop=True)
```

- Mendefinisikan dataset untuk pytorch agar bisa menggunakan dataloader

```
# Membuat dataset tensor pytorch
class SynopsisDataset(Dataset):
 def __init__(self, df):
 self.X = torch.Tensor(df['Embeddings'])
 self.y = torch.Tensor(df['Genre'])

 def __len__(self):
 return self.y.shape[0]

 def __getitem__(self, idx):
 return self.X[idx], self.y[idx]

train_dataset = SynopsisDataset(train_ds)
test_dataset = SynopsisDataset(test_ds)
```

Model

- Mendefinisikan neural network untuk klasifikasi genre

```
# Definisi classification neural network
class ClassificationNN(nn.Module):
 def __init__(self, input_size, num_classes):
 super().__init__()
 self.dense1 = nn.Linear(input_size, 1024)
 self.dense2 = nn.Linear(1024, 1024)
 self.dense3 = nn.Linear(1024, 1024)
 self.dense4 = nn.Linear(1024, num_classes)

 def forward(self, x):
 x = torch.relu(self.dense1(x))
 x = torch.relu(self.dense2(x))
 x = torch.relu(self.dense3(x))
 x = torch.sigmoid(self.dense4(x))
 return x
```

- Menginstansiasikan data loader pytorch

```
# Hyperparameter model
input_size = 4096
batch_size = 100
num_classes = 9
learning_rate = 1e-3
epochs = 20

# Data loader untuk model
train_load = DataLoader(train_dataset, batch_size, shuffle=True)
test_load = DataLoader(test_dataset, batch_size, shuffle=True)
```

Model

- Menginstansiasikan neural network klasifikasi

```
# Instansiasi
cmodel = ClassificationNN(input_size, num_classes)
cmodel
```

```
==========================================================================
ClassificationNN(  (dense1): Linear(in_features=4096, out_features=1024, bias=True)
 (dense2): Linear(in_features=1024, out_features=1024, bias=True)  (dense3):
Linear(in_features=1024, out_features=1024, bias=True)  (dense4):
Linear(in_features=1024, out_features=9, bias=True) )
```

- Menentukan perhitungan loss dan optimizer yang akan digunakan

```
# Rumus loss dan optimizer yang digunakan
loss = nn.BCELoss()
optimizer = optim.Adam(cmodel.parameters(), lr=learning_rate)
```

Training Model

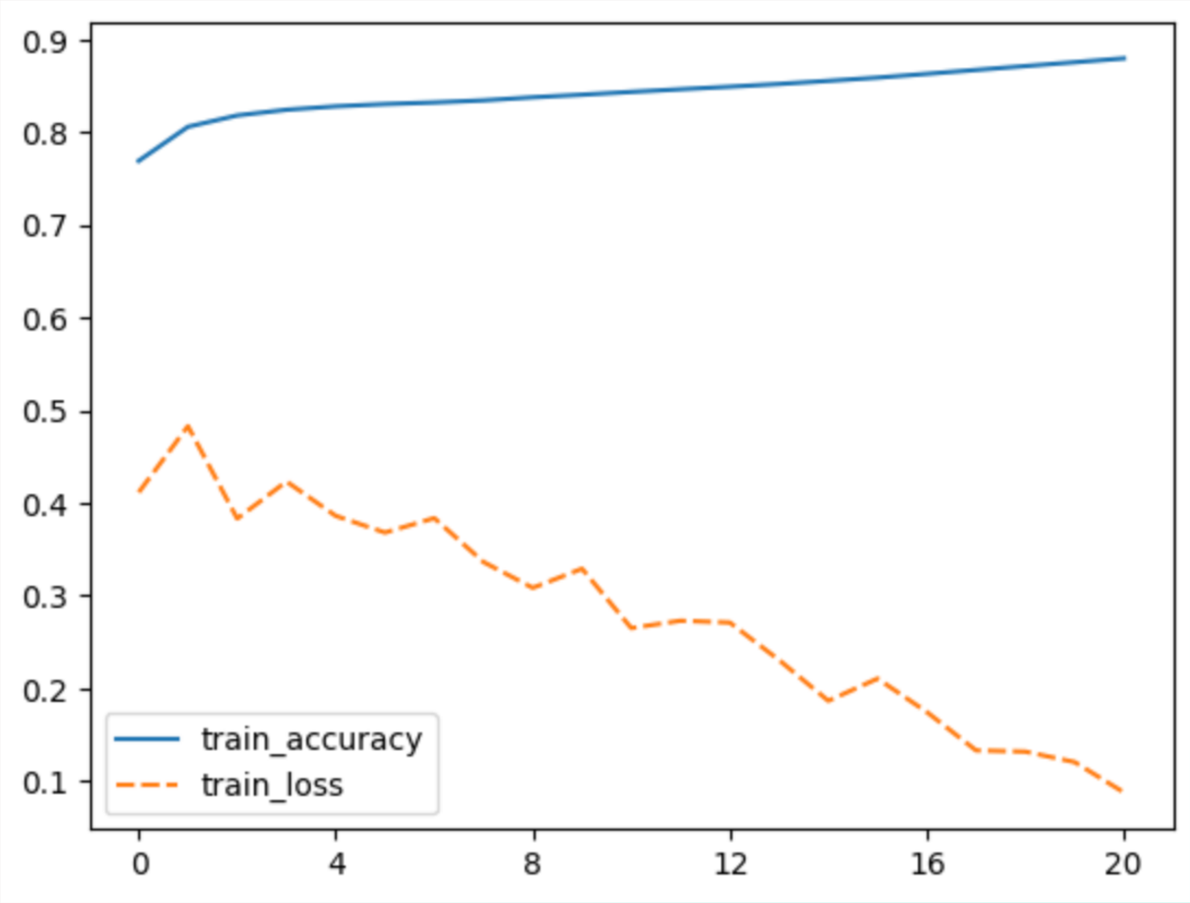
```
# Metrik akurasi training
accuracy = BinaryAccuracy()
history = {'accuracy': [], 'loss': []}
# Training model
for epoch in range(epochs+1):
 for batch_num, (synopses, targets) in enumerate(train_load):
 # Forward. Predict and calculate loss
 scores = cmodel(synopses)
 losses = loss(scores, targets)
 for (output, target) in zip(scores, targets):
 accuracy.update(output, target)
 # Backward propagration
 optimizer.zero_grad()
 losses.backward()
 optimizer.step()
 acc = accuracy.compute().item()
 print(f'epoch : {epoch}\tloss: {losses.data}\taccuracy : {acc}')
 history['accuracy'].append(acc)
 history['loss'].append(losses.data.item())
```

```
epoch : 0       loss: 0.41140255331993103       accuracy : 0.7690288424491882
epoch : 1       loss: 0.48275795578956604       accuracy : 0.8057742714881897
epoch : 2       loss: 0.38308796286582947       accuracy : 0.8180227279663086
epoch : 3       loss: 0.4230937063694   accuracy : 0.8241469860076904
epoch : 4       loss: 0.3860051929950714        accuracy : 0.8278214931488037
epoch : 5       loss: 0.3679950535297394        accuracy : 0.8302712440490723
epoch : 6       loss: 0.38357794284820557       accuracy : 0.8320209980010986
epoch : 7       loss: 0.33621516823768616       accuracy : 0.8342629075050354
epoch : 8       loss: 0.30834275484085083       accuracy : 0.8375619649887085
epoch : 9       loss: 0.3289795517921448        accuracy : 0.8403761982917786
epoch : 10      loss: 0.2649426758289337        accuracy : 0.8433945775032043
epoch : 11      loss: 0.2728864550590515        accuracy : 0.8462501168251038
epoch : 12      loss: 0.2709420323371887        accuracy : 0.8489916324615479
epoch : 13      loss: 0.2304953783750534        accuracy : 0.8519455790519714
epoch : 14      loss: 0.18650272488594055       accuracy : 0.8553028106689453
epoch : 15      loss: 0.21044743061065674       accuracy : 0.8587780594825745
epoch : 16      loss: 0.17438846826553345       accuracy : 0.8628737330436707
epoch : 17      loss: 0.13298839330673218       accuracy : 0.8670895099639893
epoch : 18      loss: 0.13165800273418427       accuracy : 0.8712605834007263
epoch : 19      loss: 0.12040550261735916       accuracy : 0.8754082918167114
epoch : 20      loss: 0.08766324818134308       accuracy : 0.8795844912528992
```

Training Model

- grafik perubahan akurasi dan loss pada saat training

```
data = {'train_accuracy': history['accuracy'], 'train_loss': history['loss']}
plot = sb.lineplot(data).set_xticks(range(0,len(history['accuracy']),int(len(history['accuracy'])/5)))
```



Result

- Hasil prediksi model klasifikasi

```
def predict(synopses, tolerance=0.3):
 synopses = model.encode(synopses, bsize=128, tokenize=False, verbose=True)
 result = cmodel(torch.Tensor(synopses))
 genres = []
 for tensor in result:
 for idx ,dim in enumerate(tensor):
 if dim > tolerance: genres.append(GENRES[idx])
 print(dim)
 return genres

predict(['''As a shocking truth about a couple's families emerges, the two lovers discover they are not so different from each other. Tessa is no longer the sweet, simple girl she was when she met Hardin.'''])
```

```
==========================================================================
Nb words kept : 32/37 (86.5%)
Speed : 3.6 sentences/s (cpu mode, bsize=128)
tensor(0.9131, grad_fn=<UnbindBackward0>)
tensor(0.9369, grad_fn=<UnbindBackward0>)
tensor(4.9106e-05, grad_fn=<UnbindBackward0>)
tensor(1.0867e-08, grad_fn=<UnbindBackward0>)
tensor(4.8591e-05, grad_fn=<UnbindBackward0>)
tensor(0.9373, grad_fn=<UnbindBackward0>)
tensor(0.0034, grad_fn=<UnbindBackward0>)
tensor(0.0040, grad_fn=<UnbindBackward0>)
tensor(0.0978, grad_fn=<UnbindBackward0>)

['Drama', 'Comedy', 'Romance']
```

Thanks