Nama : Michael Christopher
NIM : 1103210260
Analisis UTS DL Soal #1

Analisis Aspek Teknis Model MLP

1. Mengatasi Underfitting pada Model MLP

Jika model MLP dengan 3 hidden layer (256-128-64) menghasilkan underfitting pada dataset, beberapa modifikasi arsitektur yang dapat dilakukan antara lain:

Modifikasi arsitektur yang disarankan:

a) Meningkatkan kapasitas model dengan layer yang lebih lebar

- Ubah arsitektur menjadi (512-256-128) atau lebih besar
- **Alasan**: Underfitting terjadi ketika model memiliki bias tinggi dan variance rendah. Model tidak memiliki cukup kapasitas untuk menangkap pola kompleks dalam data. Dengan menambah jumlah neuron, kita meningkatkan kemampuan model untuk mempelajari representasi yang lebih kompleks.

b) Menambah kedalaman model

- Ubah menjadi (256-256-128-128-64)
- **Alasan**: Dengan menambah layer, model dapat mempelajari hierarki fitur yang lebih kompleks. Setiap layer tambahan memungkinkan model untuk menangkap abstraksi tingkat yang lebih tinggi dari data.

c) Mengubah fungsi aktivasi

- Menggunakan fungsi aktivasi yang lebih ekspresif seperti Leaky ReLU atau Swish daripada ReLU standar
- **Alasan**: Fungsi aktivasi yang lebih ekspresif dapat membantu model menangkap pola non-linear yang lebih kompleks dalam data.

d) Mengurangi regularisasi

- Mengurangi atau menghilangkan dropout dan L1/L2 regularization
- **Alasan**: Regularisasi yang terlalu kuat dapat membatasi kapasitas model. Dalam kasus underfitting, kita perlu memberikan model lebih banyak kebebasan untuk belajar.

e) Menambahkan residual/skip connections

- Implementasi skip connections antara layer
- **Alasan**: Residual connections membantu informasi mengalir lebih baik melalui network dan mengatasi masalah vanishing gradient, memungkinkan training network yang lebih dalam.

Pertimbangan Bias-Variance Tradeoff:

- **Underfitting** terjadi ketika model memiliki **bias tinggi** dan **variance rendah** - model terlalu sederhana untuk menangkap pola dalam data
- Modifikasi di atas bertujuan mengurangi bias dengan meningkatkan kapasitas model, dengan risiko meningkatkan variance
- Keseimbangan harus dijaga - jika modifikasi terlalu agresif, kita mungkin berakhir dengan overfitting (bias rendah, variance tinggi)
- Pendekatan bertahap disarankan: mulai dengan model yang lebih kapasitif, kemudian tambahkan kembali regularisasi sesuai kebutuhan sambil memantau performa pada validation set

2. Alternatif Loss Function selain MSE

Alternatif Loss Function dan Perbandingannya:

a) Mean Absolute Error (MAE)

- **Formula**: MAE = (1/n) Σ|y_true - y_pred|
- **Kelebihan**:
    - Lebih robust terhadap outlier dibanding MSE
    - Error dalam satuan yang sama dengan target (lebih mudah diinterpretasi)
- **Kekurangan**:
    - Tidak differentiable di titik nol (dapat diatasi dengan optimizers modern)
    - Converges lebih lambat dibanding MSE
- **Situasi unggul**: Dataset dengan banyak outlier; ketika kita ingin meminimalkan error absolut daripada error kuadratik

b) Huber Loss

- **Formula**:
    - $L(y, f(x)) = 0.5(y - f(x))^2$ jika $|y - f(x)| \leq \delta$
    - $L(y, f(x)) = \delta(|y - f(x)| - 0.5\delta)$ jika $|y - f(x)| > \delta$
- **Kelebihan**:
    - Kombinasi yang baik antara MSE dan MAE
    - Robust terhadap outlier seperti MAE
    - Tetap differentiable di semua titik
- **Kekurangan**:
    - Membutuhkan tuning parameter delta
- **Situasi unggul**: Dataset dengan sejumlah outlier tetapi tetap ingin konvergensi yang lebih cepat

c) Log-cosh Loss

- **Formula**: Log(cosh(y_pred - y_true))
- **Kelebihan**:
    - Menyerupai MSE untuk error kecil tetapi seperti MAE untuk error besar
    - Smooth everywhere dan memiliki derivatif kedua

    - Robust terhadap outlier
- **Kekurangan**:
    - Komputasi sedikit lebih mahal
- **Situasi unggul**: Ketika robustness terhadap outlier diperlukan dengan derivatif kedua untuk optimizers seperti L-BFGS

d) Quantile Loss

- **Formula**:
    - q(y_true - y_pred) jika y_true ≥ y_pred
    - (1-q)(y_pred - y_true) jika y_true < y_pred
- **Kelebihan**:
    - Memungkinkan prediksi interval (quantile regression)
    - Bisa diatur untuk penalizing underprediction vs overprediction
- **Kekurangan**:
    - Lebih kompleks untuk diimplementasikan dan diinterpretasi
- **Situasi unggul**: Ketika cost antara underprediction dan overprediction tidak simetris; ketika prediksi interval dibutuhkan

Rekomendasi:

Untuk dataset dengan distribusi target yang memiliki outlier signifikan, Huber Loss atau Log-cosh Loss mungkin lebih unggul daripada MSE, karena keduanya menggabungkan kecepatan konvergensi MSE dengan robustness MAE. Jika ada asymmetric penalty (misal, overprediction lebih berbahaya daripada underprediction), Quantile Loss bisa menjadi pilihan terbaik.

3. Pengaruh Perbedaan Range Nilai Fitur pada Pelatihan MLP

Fitur dengan range 0-1 versus fitur dengan range 100-1000 memiliki dampak signifikan pada proses training MLP:

Mekanisme Matematis yang Terpengaruh:

a) Komputasi Forward Pass

- Fitur dengan nilai 100-1000 akan menghasilkan pre-aktivasi ($z = wx + b$) yang jauh lebih besar
- Formula: $z_j = \Sigma(w_{ij} * x_i) + b_j$
- Dengan x_i yang bernilai besar, z_j akan didominasi oleh fitur tersebut
- Akibatnya, neuron menjadi lebih sensitif terhadap perubahan dalam fitur dengan nilai besar

b) Perhitungan Gradien

- Untuk weight w_{ij}, gradien dihitung sebagai: $\partial L/\partial w_{ij} = \partial L/\partial z_j * x_i$
- Fitur dengan range 100-1000 akan menghasilkan gradien yang 100-1000x lebih besar
- Gradien untuk fitur dengan range 0-1 akan tenggelam dibandingkan dengan fitur range besar

c) Weight Updates

- Formula: w_ij_new = w_ij_old - learning_rate * ∂L/∂w_ij
- Weights yang terhubung ke fitur bernilai besar akan mengalami update yang jauh lebih besar
- Ini menyebabkan ketidakseimbangan dalam proses pembelajaran - model akan mengoptimasi fitur berskala besar terlebih dahulu dan mungkin mengabaikan fitur berskala kecil

d) Optimasi dan Konvergensi

- Learning rate optimal untuk fitur bernilai kecil mungkin terlalu besar untuk fitur bernilai besar
- Fitur bernilai besar menyebabkan oscillation dan instabilitas dalam proses training
- Dapat menyebabkan numerical issues seperti exploding gradients
- Loss surface menjadi highly elongated dan sulit untuk dioptimasi

Dampak pada Model:

- Model akan memberikan "perhatian" yang tidak proporsional pada fitur dengan range besar
- Fitur dengan range kecil (0-1) mungkin praktis diabaikan sampai fitur range besar telah konvergen
- Konvergensi lebih lambat dan kurang stabil
- Performa model suboptimal karena tidak memanfaatkan semua informasi dengan sama baiknya

Solusi:

Normalisasi/standarisasi fitur sangat penting untuk mengatasi masalah ini. Dengan mentransformasi semua fitur ke range yang sebanding (biasanya mean 0, std 1), model dapat mempelajari semua fitur dengan tingkat "perhatian" yang seimbang.

4. Mengukur Kontribusi Relatif Fitur Tanpa Mengetahui Namanya

Untuk mengukur kontribusi relatif setiap fitur terhadap prediksi model tanpa mengetahui nama fitur, beberapa metode teknikal yang dapat digunakan:

Metode Teknikal:

a) Permutation Importance

- **Mekanisme**: Acak nilai satu fitur secara random, ukur penurunan performa, ulangi untuk semua fitur
- **Implementasi**:
    - Hitung baseline performance pada data asli
    - Untuk setiap fitur, acak nilainya (breaking relationship with target)
    - Ukur penurunan performa
    - Fitur dengan penurunan performa terbesar adalah yang paling penting

- **Kelebihan**: Model-agnostic, mudah diinterpretasi, mengukur pentingnya fitur dalam konteks model
- **Keterbatasan**:
    - Tidak menangkap interaksi antar fitur
    - Sensitif terhadap feature correlations
    - Tidak selalu konsisten jika diulang (random permutation)

b) SHAP (SHapley Additive exPlanations) Values

- **Mekanisme**: Menggunakan konsep teori game untuk mengattribusikan kontribusi setiap fitur
- **Implementasi**:
    - Ukur kontribusi marjinal setiap fitur dalam semua kemungkinan subset fitur
    - Rata-ratakan kontribusi marjinal untuk mendapatkan SHAP value
- **Kelebihan**:
    - Mempertimbangkan interaksi antar fitur
    - Solid theoretical foundation
    - Memberikan local dan global interpretability
- **Keterbatasan**:
    - Komputasi intensif (terutama untuk dataset dengan banyak fitur)
    - Asumsi feature independence yang tidak selalu valid

c) Integrated Gradients

- **Mekanisme**: Mengukur gradien model terhadap setiap fitur, diintegrasikan sepanjang path dari baseline ke instance
- **Implementasi**:
    - Pilih baseline (biasanya vektor nol)
    - Hitungkan integral gradien sepanjang straight-line path dari baseline ke instance
- **Kelebihan**:
    - Bekerja baik dengan model kompleks seperti deep neural networks
    - Memiliki axiomatic justification
- **Keterbatasan**:
    - Sensitif terhadap pilihan baseline
    - Computational overhead
    - Sulit diinterpretasi untuk non-technical audience

d) Feature Ablation

- **Mekanisme**: Menghapus satu fitur, melatih ulang model, mengukur penurunan performa
- **Implementasi**:
    - Train model dengan semua fitur (baseline)
    - Untuk setiap fitur, train model baru tanpa fitur tersebut
    - Ukur penurunan performa
- **Kelebihan**:
    - Konsep yang mudah dipahami
    - Secara langsung mengukur kontribusi fitur terhadap performa model
- **Keterbatasan**:
    - Komputasi sangat intensif (perlu melatih model sebanyak jumlah fitur)
    - Tidak menangkap interaksi kompleks antar fitur

- Hasilnya dapat dipengaruhi oleh randomness dalam proses pelatihan

Rekomendasi Penggunaan:

Kombinasi dari permutation importance (untuk initial screening yang efisien) dengan SHAP values (untuk analisis mendalam pada fitur-fitur penting) sering memberikan hasil terbaik. Untuk model neural network, integrated gradients juga layak dipertimbangkan karena secara khusus dirancang untuk deep models.

5. Desain Eksperimen untuk Memilih Learning Rate dan Batch Size Optimal

Desain Eksperimen:

A. Learning Rate

1. Learning Rate Range Test (Smith Technique)

- **Metodologi**:
  - Mulai dari learning rate sangat kecil (e.g., 1e-7)
  - Tingkatkan learning rate secara eksponensial setiap batch/iteration (hingga 1 atau lebih)
  - Plot loss vs. learning rate (log scale)
  - Optimal LR: nilai tepat sebelum loss mulai meningkat drastis

Implementasi:
```
python
lr_finder = LRFinder(model, optimizer, criterion)
lr_finder.range_test(train_loader, start_lr=1e-7, end_lr=1, num_iter=100)
lr_finder.plot() # Visualize
```

- optimal_lr = lr_finder.suggest_lr() # *Automatic suggestion*

- **Trade-off**:
  - Komputasi: Relatif ringan, selesai dalam 1 epoch
  - Kualitas hasil: Sangat baik sebagai starting point

2. Grid Search / Random Search dengan Cross-Validation

- **Metodologi**:
  - Tentukan range learning rate (e.g., [1e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3])
  - Lakukan K-fold cross-validation untuk setiap learning rate
  - Pilih learning rate dengan validation performance terbaik

Implementasi:
```
python
from sklearn.model_selection import GridSearchCV
param_grid = {'learning_rate': [1e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3]}
```

```
grid_search = GridSearchCV(estimator=model_cv_wrapper, param_grid=param_grid, cv=5)
```

- `grid_search.fit(X_train, y_train)`

- **Trade-off**:
  - Komputasi: Sangat intensif (training model sebanyak k×n kali, k=jumlah fold, n=jumlah learning rate)
  - Kualitas hasil: Sangat baik untuk generalization performance

3. Learning Rate Schedules Experiment

- **Metodologi**:
  - Setelah menemukan LR awal optimal, eksperimen dengan berbagai schedules:
    - Step decay (turun 10x setiap n epochs)
    - Exponential decay
    - Cosine annealing
    - One-cycle policy
  - Evaluasi final performance dan convergence speed
- **Trade-off**:
  - Komputasi: Moderat (satu full training per schedule)
  - Kualitas hasil: Membantu menemukan training regime optimal

B. Batch Size

1. Memory-bounded Approach

- **Metodologi**:
  - Mulai dari batch size terbesar yang dapat ditampung GPU/RAM
  - Coba batch sizes yang lebih kecil (powers of 2: 512, 256, 128, 64, 32)
  - Evaluasi validation performance dan convergence speed

Implementasi:
```
python
batch_sizes = [512, 256, 128, 64, 32]
results = {}
for bs in batch_sizes:
    train_loader = DataLoader(train_dataset, batch_size=bs)
    model, history = train_model(...)
```

- `    results[bs] = evaluate_model(...)`

- **Trade-off**:
  - Komputasi: Relatif cepat, batch size besar mengurangi jumlah iterations
  - Kualitas hasil: Batch size kecil sering meningkatkan generalization

2. Learning Rate - Batch Size Relationship

- **Metodologi**:

    - Saat meningkatkan batch size n kali, tingkatkan learning rate √n kali (linear scaling rule)
    - Evaluasi berbagai kombinasi (LR, batch size) menggunakan rule ini

Implementasi:
```
python
base_lr = 0.001
base_bs = 32
```

- `combinations = [(base_lr * (bs/base_bs)**0.5, bs) for bs in [32, 64, 128, 256, 512]]`

- **Trade-off**:
    - Komputasi: Moderat (satu training per kombinasi)
    - Efisiensi: Batch size besar dengan LR yang tepat dapat mencapai akurasi sama dengan batch kecil

3. Gradient Noise Scale Analysis

- **Metodologi**:
    - Eksperimen dengan batch size sangat kecil (e.g., 2, 4, 8)
    - Ukur variance gradien antar batch
    - Pilih batch size di mana noise gradien mulai stabil
- **Trade-off**:
    - Komputasi: Intensif, membutuhkan tracking gradien
    - Insight: Memberikan pemahaman tentang "signal-to-noise ratio" dalam gradien

Analisis Trade-off Komputasi vs Stabilitas:

1. Komputasi vs Stabilitas untuk Learning Rate

- **LR besar**:
    - Komputasi: Lebih cepat converge (potentially)
    - Stabilitas: Resiko divergence atau oscillation
- **LR kecil**:
    - Komputasi: Training lebih lambat, mungkin stuck di local minima
    - Stabilitas: Training lebih stabil, gradual improvement

2. Komputasi vs Stabilitas untuk Batch Size

- **Batch Size besar**:
    - Komputasi: Lebih efisien (better parallelization), iterations lebih sedikit
    - Stabilitas: Estimasi gradien lebih stabil, tetapi generalisasi lebih buruk
- **Batch Size kecil**:
    - Komputasi: Iterations lebih banyak, overhead lebih besar
    - Stabilitas: Noise dalam gradien, tetapi dapat membantu escape local minima dan meningkatkan generalisasi

3. Rekomendasi Optimal

- Mulai dengan learning rate range test untuk menemukan ballpark optimal LR
- Gunakan batch size terbesar yang masih memberikan validation performance baik
- Pertimbangkan square root scaling rule untuk batch size dan LR
- Evaluasi terutama pada validation set, bukan hanya training speed
- Untuk production, trade-off slight decrease in accuracy untuk significant improvement in training speed mungkin dapat diterima

Pendekatan sistematis ini membantu menemukan parameter optimal sambil menyeimbangkan trade-off antara computational efficiency dan model performance.