Nama: Azmi Abdurrahman Kautsar

NIM: 1103223211

Analisa – RegresiUTSTelkom

1. **Jika menggunakan model MLP dengan 3 hidden layer (256-128-64) menghasilkan underfitting pada sebuah dataset, modifikasi apa yang akan dilakukan pada arsitektur? Jelaskan alasan setiap perubahan dengan mempertimbangkan bias-variance tradeoff!**

   Jika model MLP dengan 3 hidden layer (256-128-64) menghasilkan underfitting, berarti model terlalu sederhana untuk menangkap kompleksitas data. Modifikasi arsitektur yang dapat dilakukan beserta alasan terkait bias-variance tradeoff adalah:

   - **Menambah Jumlah Neuron per Layer**
     Contoh: Ubah dari 256-128-64 menjadi 512-256-128.
     Alasan: Menambah kapasitas model dapat mengurangi bias, karena model menjadi lebih kompleks dan bisa mempelajari pola yang lebih rumit.
   - **Menambah Jumlah Hidden Layer**
     Contoh: Tambahkan layer menjadi 4 atau 5 (misal: 512-256-128-64-32).
     Alasan: Kedalaman tambahan memungkinkan model membentuk representasi hierarkis, membantu menangkap struktur kompleks dan juga mengurangi bias.
   - **Mengganti Aktivasi atau Menambahkan Nonlinearitas**
     Contoh: Gunakan ReLU jika belum, atau eksplorasi seperti LeakyReLU.
     Alasan: Fungsi aktivasi yang tepat meningkatkan kemampuan model dalam memodelkan hubungan nonlinier, mengurangi bias.
   - **Kurangi Regularisasi (Dropout/L2)**
     Contoh: Jika dropout terlalu tinggi (misal 0.5), kurangi jadi 0.2.
     Alasan: Regularisasi mencegah overfitting, tapi terlalu kuat bisa membuat model tidak belajar cukup → terlalu bias.
   - **Latih Lebih Lama atau Gunakan Learning Rate yang Lebih Baik**
     Contoh: Tambah lagi epochnya.
     Alasan: Underfitting bisa juga karena model belum cukup belajar. Training lebih lama dapat mengurangi bias, tapi hati-hati jangan sampai overfit.

   Untuk mengatasi underfitting, tingkatkan kompleksitas model (jumlah layer/neuron) dan kurangi hambatan belajar (seperti regularisasi berlebih). Ini akan mengurangi bias, meskipun variance bisa meningkat. Namun karena awalnya underfit, peningkatan variance masih dalam batas aman.

2. **Selain MSE, loss function apa yang mungkin cocok untuk dataset ini? Bandingkan kelebihan dan kekurangannya, serta situasi spesifik di mana alternatif tersebut lebih unggul daripada MSE!**

   Selain Mean Squared Error (MSE), beberapa loss function lain yang cocok untuk dataset besar tergantung pada jenis data dan tujuan. Berikut beberapa alternatif, lengkap dengan kelebihan, kekurangan, dan kapan lebih baik dari MSE:

   - **Mean Absolute Error (MAE)**
     - Kelebihan: Lebih robust terhadap outlier dibanding MSE, memberi penalti linier (bukan kuadratik seperti MSE).
     - Kekurangan: Gradient-nya konstan → bisa menyebabkan pelatihan lambat, tidak smooth di titik 0 (turunan tidak terdefinisi).
     - Lebih unggul dari MSE ketika: Data mengandung banyak outlier, akurasi pada mayoritas data lebih penting daripada penalti besar pada error besar.
   - **Huber Loss**
     - Kelebihan: Kompromi antara MSE (kuat untuk data normal) dan MAE (tahan outlier), smooth dan stabil saat training.
     - Kekurangan: Perlu memilih parameter & secara tepat.
     - Lebih unggul dari MSE ketika: Dataset besar dengan beberapa outlier, tapi tetap ingin mempertahankan sensitivitas MSE pada error kecil.
   - **Log-Cosh Loss**
     - Kelebihan: Mirip MSE untuk error kecil, mirip MAE untuk error besar, smooth dan differentiable di seluruh domain.
     - Kekurangan: Sedikit lebih mahal secara komputasi dibanding MSE.
     - Lebih unggul dari MSE ketika: Dataset besar dan ingin loss function smooth tapi tahan terhadap outlier.
   - **Quantile Loss**
     - Kelebihan: Cocok untuk data heteroskedastik (distribusi error tidak seragam), dapat memodelkan interval prediksi.
     - Kekurangan: Tidak cocok untuk regresi biasa.
     - Lebih unggul dari MSE ketika: Ingin estimasi distribusi output, bukan hanya prediksi rata-rata.

3. **Jika salah satu fitur memiliki range nilai 0-1, sedangkan fitur lain 100-1000, bagaimana ini memengaruhi pelatihan MLP? Jelaskan mekanisme matematis (e.g., gradien, weight update) yang terdampak!**

Perbedaan skala antar fitur (misalnya 0–1 vs. 100–1000) dapat mengganggu pelatihan MLP, terutama pada mekanisme propagasi dan update bobot. Berikut penjelasan matematisnya:

- **Pengaruh pada Aktivasi**
  - Input x ke neuron dihitung sebagai:
    $$z = w_1x_1 + w_2x_2 + \ldots + b$$
  - Jika $x_1 \in [0,1]$ dan $x_2 \in [100,1000]$, maka x2 akan dominan, menyebabkan z bergantung terutama pada x2, dan x1 hampir diabaikan.
- **Pengaruh pada Gradien**
  - Backpropagation menghitung turunan loss terhadap bobot:
    $$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial w_i} = \delta \cdot x_i$$
  - Jika xi besar (misal 1000), maka gradien untuk wi jadi sangat besar → menyebabkan update bobot besar → training tidak stabil.
  - Sebaliknya, fitur kecil (0–1) → gradien kecil → belajar lambat.
- **Masalah pada Optimisasi**
  - Landscape dari fungsi loss menjadi tidak seimbang (anisotropic), sehingga: Gradient descent lambat (zigzag) dan bisa gagal konvergen atau butuh learning rate kecil → pelatihan jadi lambat.

Solusi: Normalisasi atau Standardisasi

- Min-Max Scaling (0–1) atau Standardization (mean 0, std 1) menyetarakan skala semua fitur.
- Membantu: Gradien jadi seimbang, konvergensi lebih cepat, semua fitur ikut berkontribusi.

Perbedaan skala antar fitur menyebabkan dominan satu fitur, gradien tidak seimbang, dan update bobot yang tidak efisien, yang semuanya menghambat pelatihan. Normalisasi wajib untuk performa optimal MLP.

4. **Tanpa mengetahui nama fitur, bagaimana anda mengukur kontribusi relatif setiap fitur terhadap prediksi model? Jelaskan metode teknikal (e.g., permutation importance, weight analysis) dan keterbatasannya!**

Berikut beberapa metode teknikal untuk mengukur kontribusi relatif fitur terhadap prediksi model (seperti MLP), tanpa mengetahui nama fitur:

- Permutation Feature Importance
  - Cara kerja:

        - Acak (shuffle) satu fitur pada input, ukur penurunan performa (misal akurasi, MSE).
        - Semakin besar penurunan, semakin penting fitur tersebut.
    - Kelebihan:
        - Agnostik terhadap model (bisa untuk MLP, tree, dll).
        - Tidak butuh akses ke struktur model.
    - Keterbatasan:
        - Mahal secara komputasi.
        - Fitur yang terkorelasi bisa salah dinilai (shuffling satu fitur bisa dipengaruhi fitur lain).
- SHAP (SHapley Additive exPlanations)
    - Cara kerja:
        - Berdasarkan teori permainan, menghitung kontribusi setiap fitur pada setiap prediksi.
        - Nilai SHAP = kontribusi marjinal rata-rata fitur terhadap output model.
    - Kelebihan:
        - Konsisten dan secara teori kuat.
        - Bisa menunjukkan kontribusi fitur secara lokal dan global.
    - Keterbatasan:
        - Sangat mahal untuk model besar (seperti deep neural nets).
        - Sulit dihitung langsung tanpa tool/library (misal shap di Python).
- Weight Magnitude Analysis (untuk MLP)
    - Cara kerja:
        - Lihat bobot lapisan pertama: fitur input → hidden layer.
        - Hitung rata-rata atau norma bobot tiap fitur input.
    - Kelebihan:
        - Sangat cepat dan langsung.
        - Bisa dilakukan saat model sudah dilatih.
    - Keterbatasan:
        - Tidak memperhitungkan non-linearitas (aktivasi, hidden layer).
        - Tidak selalu mencerminkan pengaruh sebenarnya (terutama pada deep model).
- Input Gradient (Saliency Maps)
    - Cara kerja:
        - Hitung turunan output terhadap input ($\partial y/\partial x$).
        - Menunjukkan seberapa sensitif prediksi terhadap perubahan kecil pada input fitur.
    - Kelebihan:
        - Akurat untuk melihat sensitivitas lokal.

        - Digunakan dalam interpretasi CNN dan MLP.
    - Keterbatasan:
        - Hanya lokal (satu instance), bukan global importance.
        - Bisa noisy dan sulit diinterpretasi secara umum.

5. **Bagaimana cara mendesain eksperimen untuk memilih learning rate dan batch size secara optimal? Sertakan analisis tradeoff antara komputasi dan stabilitas pelatihan!**

Berikut cara mendesain eksperimen untuk memilih learning rate dan batch size secara optimal dengan analisis trade-off antara komputasi dan stabilitas pelatihan:

- Grid Search atau Random Search
    - Langkah:
        - Pilih rentang:
            - Learning rate: $[10^{-5}, 10^{-1}]$ (log-scale)
            - Batch size: $[16, 32, 64, 128, 256]$
        - Jalankan eksperimen kombinasi (grid) atau sampling acak (random search).
    - Kelebihan: Mudah dan paralel.
    - Kekurangan: Mahal secara komputasi.

- Learning Rate Finder (1cycle method)
    - Langkah:
        - Mulai training dengan LR sangat kecil dan naikkan secara eksponensial.
        - Plot loss vs LR.
        - Pilih LR di titik sebelum loss mulai meningkat tajam.
    - Kelebihan: Efisien dan sering akurat.
    - Kekurangan: Hanya untuk LR, tidak batch size.
- Learning Rate vs Batch Size Scaling (Empirical Rule)
    - Ada korelasi empiris: learning rate $\propto$ batch size
    - Saat batch size besar, learning rate juga bisa ditingkatkan agar training tetap cepat dan stabil.

- Monitoring Metrics
    - Evaluasi setiap kombinasi berdasarkan:
        - Validation loss/accuracy
        - Stabilitas gradien (tidak exploding/vanishing)
        - Waktu per epoch dan jumlah epoch hingga konvergen