Nama : Muhammad Husain Kaasyiful Ghitha

NIM : 1103220047

Kelas TK-45-G13

Deep Learning

Jawaban Soal Analisis UTS

1. Arsitektur CNN dengan X lapisan konvolusi menghasilkan akurasi training 98% tetapi akurasi validasi 62%. Jelaskan fenomena vanishing gradient yang mungkin terjadi pada lapisan awal, dan bagaimana cara memitigasinya! Mengapa penambahan Batch Normalization setelah lapisan konvolusi ke-Y justru memperburuk generalisasi, serta strategi alternatif untuk menstabilkan pembelajaran?

   Ketika CNN dengan X lapisan konvolusi menghasilkan akurasi training 98% namun akurasi validasi hanya 62%, hal ini menunjukkan masalah overfitting yang serius. Salah satu fenomena yang dapat terjadi adalah vanishing gradient, terutama pada lapisan awal.
   Vanishing gradient terjadi ketika gradien yang digunakan untuk mengupdate parameter pada lapisan awal menjadi sangat kecil mendekati nol selama backpropagation. Secara matematis, ini disebabkan oleh efek rantai dalam chain rule:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a_n} \cdot \frac{\partial a_n}{\partial a_{n-1}} \cdot \ldots \cdot \frac{\partial a_2}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1}$$

   Dengan bertambahnya jumlah lapisan (X besar), perkalian dari derivatif parsial yang nilainya kurang dari 1 akan menghasilkan nilai yang sangat kecil untuk gradien lapisan awal.

   Strategi Mitigasi Vanishing Gradient:

   a. Penggunaan Fungsi Aktivasi yang Tepat: Jika masih menggunakan aktivasi sigmoid. Hal yang dilakukan adalah mengganti sigmoid/tanh dengan ReLU atau variannya (LeakyReLU, ELU, SELU) yang memiliki gradien 1 untuk input positif. Contohnya:

$$\mathrm{ReLU(x)} = \max(0, \mathrm{x})$$
$$\mathrm{LeakyReLU(x)} = \max(0.01\mathrm{x}, \mathrm{x})$$

   b. Skip Connections/Residual Connections: Menerapkan arsitektur ResNet yang memungkinkan gradien mengalir langsung ke lapisan sebelumnya:

$$h_{l+1} = h_l + F(h_l, W_l)$$

c. Inisialisasi Bobot yang Tepat: Menggunakan He initialization untuk ReLU atau Xavier/Glorot initialization untuk sigmoid/tanh:

$$W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{in}}}\right), \text{ untuk He init}$$

$$W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{in} + n_{out}}}\right), \text{ untuk Xavier init}$$

Batch Normalization di lapisan ke-Y dapat memperburuk generalisasi karena:

a. Internal Covariate Shift pada Mini-batch Kecil: Pada mini-batch yang terlalu kecil, statistik (mean dan variance) menjadi tidak stabil, menghasilkan normalisasi yang tidak konsisten
b. Over-dependence pada Distribusi Training: BN pada lapisan ke-Y mungkin terlalu agresif menyesuaikan dengan distribusi data training, mengurangi kemampuan generalisasi
c. Parameter Tambahan γ dan β Dapat meningkatkan kompleksitas model dan potensi overfitting.

Strategi Alternatif untuk Stabilisasi:

a. Layer Normalization: Normalisasi pada dimensi fitur, bukan batch.

$$\hat{x}_i = \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$

b. Group Normalization: Mengelompokkan channel dan melakukan normalisasi per grup.

$$\hat{x}_{i,g} = \frac{x_{i,g} - \mu_{i,g}}{\sqrt{\sigma_{i,g}^2 + \epsilon}}$$

c. Weight Normalization: Memisahkan besaran dan arah parameter.

$$w = g \cdot \frac{v}{||v||}$$

2. Ketika melatih CNN dari nol, loss training stagnan di nilai tinggi setelah XXX(3 digit epoch) epoch. Identifikasi tiga penyebab potensial terkait laju pembelajaran (learning rate), inisialisasi berat, atau kompleksitas model! Mengapa penggunaan Cyclic Learning Rate dapat membantu model keluar dari local minima, dan bagaimana momentum pada optimizer SGD memengaruhi konvergensi?

   Kemungkinan penyebab:

   a. Learning Rate yang Tidak Tepat:
      - Terlalu Besar: Menyebabkan osilasi atau divergensi karena step terlalu besar.

        $$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t)$$

        Jika α terlalu besar, θ akan melewati minimum
      - Terlalu Kecil: Konvergensi sangat lambat atau terjebak di plateau $\lim_{n\to\infty} \left|\left|\theta_{t+1} - \theta_t\right|\right| \approx 0$ meskipun $\left|\left|\nabla J(\theta_t)\right|\right| > \epsilon$.

   b. Inisialisasi Bobot yang Buruk:
      - Terlalu Besar: Menyebabkan saturasi pada fungsi aktivasi, menghasilkan gradien mendekati nol.
      - Terlalu Kecil: Signal mengecil saat forward pass, menyebabkan vanishing gradient.
      - Distribusi Tidak Tepat: Ketidaksesuaian antara distribusi inisialisasi dengan fungsi aktivasi.

   c. Kompleksitas Model vs Data:
      - Underfitting: Model terlalu sederhana untuk mempelajari pola kompleks.
      - Arsitektur Tidak Sesuai: Kurangnya kapasitas representasi untuk menangkap fitur penting.
      - Kurangnya Regularisasi: Model terperangkap di saddle point pada lanskap loss.

   Cyclic Learning Rate dan Local Minima:
   Cyclic Learning Rate (CLR) membantu model keluar dari local minima melalui variasi sistematis learning rate antara nilai rendah dan tinggi:

   $$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left(1 + \sin\left(\frac{2\pi t}{T}\right)\right)$$

   Dengan variasi teratur ini:

   a. Phase Eksplorasi: Learning rate tinggi mendorong model keluar dari local minima atau saddle point.
   b. Phase Eksploitasi: Learning rate rendah memungkinkan konvergensi ke minimum yang lebih baik.

c. Stochastic Resonance: Fluktuasi teratur menambah noise yang membantu mengatasi barrier energi.

Pengaruh Momentum pada SGD:
Momentum memperkenalkan term kecepatan dalam update parameter:

$$v_t = \gamma v_{t-1} + \eta \nabla J(\theta_{t-1})$$

$$\theta_t = \theta_{t-1} - v_t$$

Pengaruhnya pada konvergensi:

a. Percepatan Konvergensi: Mempercepat pembelajaran di arah konsisten.

$$\theta_t \approx \theta_{t-1} - \eta \sum_{i=0}^{t-1} \gamma^i \nabla J(\theta_{t-1-i})$$

b. Dampening Oscillation: Mengurangi osilasi pada arah dengan gradien yang berubah-ubah.
c. Overcoming Plateaus: Membantu model bergerak melewati area dengan gradien kecil $||v_t|| > \|\eta \nabla J(\theta_t)\|$ ketika di plateau.

3. Pada klasifikasi spesies ikan menggunakan CNN, penggunaan fungsi aktivasi ReLU tidak menunjukkan peningkatan akurasi setelah 50 epoch, meskipun learning rate telah dioptimasi. Jelaskan fenomena dying ReLU yang mungkin terjadi dan bagaimana hal ini mengganggu aliran gradien selama backpropagation!

Fenomena dying ReLU terjadi ketika neuron dengan aktivasi ReLU secara konsisten menghasilkan output negatif atau nol, yang menyebabkan neuron tersebut menjadi inaktif permanen.
Secara matematis, ReLU didefinisikan sebagai:

$$ReLU(x) = \max(0, x)$$

Ketika input x < 0, output ReLU = 0 dan gradiennya juga 0:

$$\frac{d}{dx} ReLU(x) = \begin{cases} 1 \text{ untuk } x > 0 \\ 0 \text{ untuk } x \leq 0 \end{cases}$$

Dying ReLU terjadi ketika:

a. Weight Update Negatif: Update bobot yang besar menyebabkan input ke ReLU selalu negatif.

$$z = \sum_i w_i x_i + b < 0$$

b. Learning Rate Terlalu Tinggi: Menyebabkan perubahan bobot yang besar dan mendorong neuron ke area negatif.

$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$$

c. Inisialisasi Bobot yang Buruk: Bobot awal yang terlalu negatif menyulitkan neuron untuk keluar dari kondisi "mati".

Dampak pada Backpropagation:
Ketika neuron "mati", aliran gradien selama backpropagation terganggu:

a. Gradient Blockage: Gradien tidak bisa mengalir melalui neuron yang "mati"

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w_i} = \frac{\partial L}{\partial a} \cdot 0 \cdot x_i = 0$$

b. Representational Capacity Loss: Kapasitas model efektif berkurang dengan setiap neuron yang "mati". Jika proporsi p dari neuron "mati", kapasitas efektif menjadi (1-p) dari kapasitas asli.
c. Sparse Activation: Aktivasi yang terlalu sparse mengurangi kekuatan representasional.

Solusi untuk Dying ReLU:

a. Leaky ReLU: Memberikan gradient kecil untuk input negatif

$$LeakyReLU(x) = \begin{cases} x \; untuk \; x > 0 \\ \alpha x \; untuk \; x \leq 0 \end{cases}$$

dimana α adalah parameter kecil (biasanya 0.01).

b. PReLU (Parametric ReLU): Slope negatif yang dapat dilatih

$$PReLU(x) = \begin{cases} x \; untuk \; x > 0 \\ \alpha x \; untuk \; x \leq 0 \end{cases}$$

dimana α adalah parameter yang dilatih.

c. ELU (Exponential Linear Unit): Memberikan saturasi negatif.

$$ELU(x) = \begin{cases} x \; untuk \; x > 0 \\ \alpha(e^x - 1) \; untuk \; x \leq 0 \end{cases}$$

4. Pada pelatihan CNN untuk klasifikasi XX spesies ikan, grafik AUC-ROC menunjukkan satu kelas (Spesies X) stagnan di 0.55 sementara kelas lain mencapai >0.85 setelah YYY epoch. Analisis mengapa class-weighted loss function gagal meningkatkan kinerja Spesies X, dan identifikasi tiga faktor penyebab potensial terkait karakteristik data dan arsitektur model!

   Dalam klasifikasi multi-kelas dengan spesies ikan, ketika grafik AUC-ROC menunjukkan satu kelas (Spesies X) stagnan di 0.55 sementara kelas lain melebihi 0.85, ini mengindikasikan ketidakseimbangan performa yang signifikan. Performa AUC-ROC yang rendah (0.55) hampir mendekati klasifikasi acak (0.5).

   Alasan Class-weighted Loss Function Gagal:
   Class-weighted loss function umumnya bekerja dengan memberikan bobot lebih tinggi pada kelas yang underrepresented:

$$L_{weighted} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} w_j \cdot y_{i,j} \log(\hat{y}_{i,j})$$

   Namun, kegagalan peningkatan kinerja dapat disebabkan oleh:

   a. Masalah Beyond Class Imbalance: Bobot hanya mengatasi frekuensi kelas, bukan kompleksitas representasi. Hasil $w_j \propto \frac{1}{frequency(j)}$ tidak menjamin $P(correct|j)$ meningkat.

   b. Optimasi Terlalu Fokus pada Loss: Model mungkin mengurangi total loss dengan fokus pada kelas mayoritas dan "mengorbankan" kelas problematik.

$$L_{total} \approx \sum_{j \neq X} L_j + \epsilon \cdot L_X$$

   c. Pemilihan Bobot yang Tidak Optimal: Bobot terlalu ekstrem dapat menyebabkan ketidakstabilan training. Jika $w_X \gg w\_j$ untuk $j \neq X$, gradien didominasi oleh kelas X.

   Tiga Faktor Penyebab Potensial:

   a. Faktor Karakteristik Data:
      - Feature Distinctiveness: Spesies X mungkin memiliki fitur visual yang sangat mirip dengan spesies lain.

$$Similarity(X,Y) = \frac{\langle f_X, f_Y \rangle}{||f_X|| \cdot ||f_Y||} \approx 1$$

- Variasi Intra-kelas Tinggi: Variasi penampilan dalam spesies X lebih besar daripada antar-spesies.

$$\sigma^2_{within(X)} > \sigma^2_{between(X,Y)}$$

- Anomali atau Noise: Dataset untuk spesies X mungkin mengandung label salah atau gambar berkualitas rendah.

Faktor Arsitektur Model:

- Representasi Fitur Tidak Memadai: Arsitektur CNN mungkin tidak mengekstrak fitur yang membedakan spesies X.

$$\exists\, f \text{ s.t } f \text{ penting untuk X tetapi } CNN(f) \approx 0$$

- Decision Boundary Kompleks: Batas keputusan untuk spesies X membutuhkan non-linearitas yang lebih tinggi.
- Kurangnya Kapasitas Model: Model terlalu sederhana untuk menangkap nuansa spesies X.

Faktor Optimasi:

- Pembelajaran yang Tidak Seimbang: Model terlalu mengoptimasi kelas mayoritas.
- Gradien yang Tidak Stabil: Gradien untuk contoh dari spesies X mungkin memiliki varians tinggi.

$$Var(\nabla L_X) \gg Var(\nabla L_j) \text{ untuk } j \neq X$$

- Premature Convergence: Model terjebak dalam local minima yang buruk untuk spesies X.

Solusi Potensial:

- Focal Loss: Memberikan bobot lebih pada contoh yang sulit diklasifikasi.

$$FL(p_t) = -(1 - p_t)^{\gamma} \log(p_t)$$

- Data Augmentasi Spesifik-Kelas: Augmentasi khusus untuk spesies X.
- Arsitektur Hibrida: Menggunakan subnetwork khusus untuk spesies yang sulit diklasifikasi.

5. Pada arsitektur CNN untuk klasifikasi ikan, peningkatan kompleksitas model justru menyebabkan penurunan akurasi validasi dari 85% ke 65%, meskipun akurasi training mencapai 98%. Jelaskan fenomena overfitting yang terjadi, dan mengapa penambahan kapasitas model tidak selalu meningkatkan generalisasi! Identifikasi 3 kesalahan desain arsitektur yang memicu degradasi performa

Ketika arsitektur CNN yang lebih kompleks menyebabkan penurunan akurasi validasi dari 85% ke 65%, meskipun akurasi training mencapai 98%, terdapat indikasi overfitting klasik yang sering terjadi. Overfitting terjadi ketika model memiliki kapasitas yang berlebih untuk "menghafal" data training daripada "memahami" pola umum.

Alasan Penambahan Kapasitas Tidak Selalu Meningkatkan Generalisasi:
Secara teoritis, performa model dapat dianggap sebagai keseimbangan antara bias dan varians:

$$\text{Error total} = \text{Bias}^2 + \text{Variance} + \text{Noise}$$

a. Peningkatan Kompleksitas Menurunkan Bias tetapi Meningkatkan Varians.
   Dengan model lebih kompleks, kapasitas fitting meningkat (menurunkan bias), tetapi ketidakpastian estimasi parameter juga meningkat (meningkatkan varians)
b. Kebutuhan Data Meningkat Secara Eksponensial dengan Kompleksitas.
   Jumlah parameter model, p, membutuhkan data dalam orde $O(p)$ untuk estimasi yang stabil
c. Curse of Dimensionality: Ruang parameter yang lebih besar menyebabkan data training menjadi relatif semakin sparse.

   $$\text{Densitas data} \propto \frac{N}{V} \propto \frac{N}{L^d}$$

   dimana N adalah jumlah sampel, L adalah lebar ruang, dan d adalah dimensi ruang parameter

Tiga Kesalahan Desain Arsitektur yang Memicu Degradasi Performa:
a. Kedalaman Berlebih Tanpa Skip Connections: Hal ini menyebabkan gradient vanishing dan exploding yang terjadi menjadi lebih parah. Setiap lapisan tambahan meningkatkan kemungkinan gradien yang mengecil secara multiplikatif.

   $$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a_n} \cdot \frac{\partial a_n}{\partial a_{n-1}} \cdot \ \ldots \cdot \frac{\partial a_2}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1}$$

   Sehingga, lapisan awal tidak diupdate secara efektif, mengurangi kapasitas belajar efektif

b. Overparameterization Tanpa Regularisasi yang Memadai: Jumlah parameter yang berlebihan tanpa constraint menghasilkan ruang solusi terlalu luas. Model dengan parameter berlebih memiliki ragam (variance) tinggi

   $$Var(\hat{f}) \propto \frac{p}{N}$$

   dimana p adalah jumlah parameter dan N adalah jumlah sampel. Akibatnya, Model menyesuaikan dengan noise pada data training

c. Desain Layer yang Tidak Efisien: Terkadang, di dalam arsitektur, terdapat penggunaan layer yang tidak tepat. Misal, pada arsitektur terlalu banyak fully-connected layer dibandingkan dengan convolutional layer. Masalahnya, Fully-connected layer memiliki parameter yang jauh lebih banyak.

$$Params_{FC} = n_{in} \times n_{out} \gg Params_{Conv} = k \times k \times c_{in} \times c_{out}$$

Dampak: Peningkatan parameter yang tidak menghasilkan extraksi fitur yang lebih baik

Mitigasi Overfitting dalam Model Kompleks:

a. Regularisasi yang Tepat: Regularisasi berguna dalam mencegah model untuk terlalu terpaku pada input yang timpang, dan meningkatkan kemampuan generalisasi model. Rumus yang dapat digunakan:
   - L1 Regularization:

$$L_{reg} = L + \lambda \sum_i |w_i|$$

   - Atau L2 Regularization:

$$L_{reg} = L + \lambda \sum_i w_i^2$$

b. Arsitektur dengan Inductive Bias yang Sesuai:
   - Residual Connections: $h_{l+1} = h_l + F(h_l, W_l)$
   - Depthwise Separable Convolutions untuk efisiensi parameter