Nama : Hamdan Syaifuddin Zuhri
NIM : 1103220220
Kelas : TK-45-G13

# UTS Deep Learning

## Classification

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?
   Jawab:
   **Analisis Arsitektur CNN & Masalah Generalisasi**
   a. Overfitting pada CNN
      Pada arsitektur CNN dengan X lapisan konvolusi, diperoleh:
      - Akurasi training: 98%
      - Akurasi validasi: 62%

      Perbedaan besar antara training dan validasi ini merupakan indikasi kuat overfitting, di mana model belajar sangat baik pada data pelatihan, tetapi gagal menggeneralisasi ke data baru.
   b. **Fenomena Vanishing Gradient**
      Vanishing gradient terjadi ketika:
      - Gradien yang dihitung saat backpropagation menjadi sangat kecil di lapisan awal (dekat input).
      - Hal ini menyebabkan lapisan awal hampir tidak belajar, meskipun sangat penting untuk mengekstraksi fitur dasar seperti tepi dan tekstur.

      Ini umum terjadi ketika:
      - Model sangat dalam (banyak layer konvolusi).
      - Aktivasi seperti sigmoid atau tanh digunakan, yang membatasi rentang output dan memperkecil gradien saat chain rule diterapkan berulang kali.

      Mitigasi Vanishing Gradient:
      - Gunakan aktivasi ReLU atau turunannya (misalnya LeakyReLU) yang tidak menyebabkan saturasi.
      - Weight initialization yang baik, seperti He initialization (sangat cocok dengan ReLU).
      - Residual connections (seperti pada ResNet) yang memungkinkan gradien mengalir langsung ke lapisan awal.

- Batch Normalization (BN) dapat membantu, tetapi tidak selalu efektif jika tidak ditempatkan secara strategis.

c. **Mengapa Batch Normalization Justru Memperburuk Generalisasi**
Penambahan Batch Normalization setelah lapisan konvolusi ke-Y mungkin justru menyebabkan:
   - Over-normalisasi terhadap fitur antara convolution dan activation (misalnya BN sebelum ReLU), membuat representasi terlalu "terskala".
   - Pada beberapa layer, BN bisa mengurangi variasi antar sampel yang sebetulnya penting untuk generalisasi.
   - Jika diterapkan terlalu dekat dengan output (misal pada shallow CNN), bisa mengganggu distribusi fitur akhir yang dibutuhkan untuk klasifikasi.

   Contoh kesalahan umum:
   Menambahkan BN di lapisan yang sebenarnya sudah cukup stabil atau tidak memerlukan normalisasi dapat menyebabkan informasi penting dari fitur justru dihapus.

d. **Strategi Alternatif untuk Menstabilkan Pembelajaran**
   Jika BN memperburuk generalisasi, alternatif lain meliputi:
   Dropout:
   - Secara acak mengabaikan neuron selama training, mencegah overfitting.
   - Baik diterapkan setelah FC layer atau setelah pooling layer.

   Layer Normalization atau Group Normalization:
   - Lebih stabil pada mini-batch kecil atau ketika batch size sangat terbatas.
   - Tidak bergantung pada statistik antar-batch seperti BN.

   Early Stopping:
   - Menghentikan pelatihan saat validasi tidak meningkat.

   Data Augmentation:
   - Menambah variasi data secara artifisial agar model belajar lebih robust.

   Smaller Architecture / Regularization:
   - Kurangi kedalaman CNN jika dataset tidak besar.
   - Tambahkan L2 regularization pada layer-layer dense.

   Kesimpulan
   - Vanishing gradient bisa menghambat pembelajaran lapisan awal dalam CNN dalam-dalam.

- Batch Normalization bisa membantu, tapi penempatan yang tidak tepat (misalnya di layer ke-Y) bisa memperburuk generalisasi.
- Gunakan kombinasi strategi seperti ReLU, weight initialization, residual connection, dropout, dan layer normalization untuk pembelajaran yang stabil dan generalisasi yang baik.

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?
   Jawab:
   **Analisis Ketika Training Loss CNN Stagnan di Nilai Tinggi**
   Selama pelatihan CNN dari nol, loss training stagnan pada nilai tinggi setelah ratusan epoch (misal: >100 epoch). Ini mengindikasikan bahwa model tidak mampu menurunkan error, bahkan pada data latih, dan proses pelatihan mengalami "buntu".
   **Penyebab Potensial:**
   a. Laju Pembelajaran (Learning Rate) Terlalu Tinggi atau Terlalu Rendah
      - Terlalu tinggi: parameter melompat-lompat di sekitar minima, tidak pernah konvergen → loss tetap tinggi.
      - Terlalu rendah: proses training sangat lambat, stuck di daerah loss tinggi karena tidak cukup tenaga untuk keluar dari dataran tinggi/plateau.

      Solusi: Gunakan Learning Rate Scheduler seperti Cyclic Learning Rate atau ReduceLROnPlateau untuk menyesuaikan LR secara dinamis.
   b. Inisialisasi Berat yang Buruk
      Jika bobot layer (khususnya konvolusi dan dense) diinisialisasi dengan nilai yang terlalu kecil atau besar:
      - Output aktivasi bisa jadi sangat kecil (vanishing) atau terlalu besar (exploding).
      - Layer awal bisa berhenti belajar (sama seperti vanishing gradient).

      Solusi:
      - Gunakan He Initialization untuk arsitektur berbasis ReLU.
      - Pastikan layer CNN tidak menggunakan default initializer sembarangan (misal: uniform dengan batas kecil).
   c. Model Terlalu Kompleks terhadap Dataset
      CNN dengan banyak layer dan parameter besar (seperti EfficientNet, ResNet) bisa:
      - Sulit dilatih dari nol tanpa pretraining jika dataset kecil/sederhana.
      - Overparameterized → menyebabkan pelatihan lambat dan stuck.

      Solusi:
      - Mulai dari model kecil atau pretrained CNN.

- Tambahkan regularisasi, dropout, dan/atau kurangi layer konvolusi.

**Peran Cyclic Learning Rate (CLR)**

Cyclic Learning Rate mengubah learning rate secara periodik, misalnya naik-turun antara min_lr dan max_lr.

Mengapa membantu?

- Ketika model terjebak di local minima atau saddle point (dataran tinggi), peningkatan LR secara siklikal: Memberi "energi" untuk keluar dari jebakan., Memungkinkan eksplorasi parameter space yang lebih luas.
- Mirip dengan simulated annealing: hindari stuck terlalu awal dalam landscape loss yang kompleks.

Contoh dalam praktik:
tf.keras.experimental.CosineDecayRestarts(...)

**Pengaruh Momentum pada SGD**

Momentum dalam optimizer SGD mengacu pada menambahkan "gaya dorong" ke arah gradien sebelumnya, agar:

- Gradien tidak hanya berdasarkan satu batch saja.
- Menghindari osilasi dan mempercepat konvergensi pada arah yang konsisten.

Bagaimana membantu:

- Jika tanpa momentum, model bisa berosilasi di sekitar titik curam tanpa arah stabil.
- Momentum tinggi (0.9–0.99) mempercepat penurunan loss dan membantu melintasi area datar (plateau).

Namun, jika terlalu besar, bisa menyebabkan overshooting → loss naik turun tanpa stabil.

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!

   Jawab:

   **Analisis Masalah: CNN Tidak Meningkat Setelah 50 Epoch (ReLU Activation)**

   Model CNN untuk klasifikasi gambar ikan tidak mengalami peningkatan akurasi **setelah 50 epoch**, meskipun **learning rate sudah dioptimasi** (misalnya via scheduler atau tuning). Fungsi aktivasi yang digunakan adalah **ReLU**.

   a. **Dugaan Utama: Dying ReLU Phenomenon**

      Apa itu Dying ReLU?

Fungsi aktivasi ReLU ($f(x) = \max(0, x)$) hanya menghasilkan output > 0 jika input-nya positif.
Jika selama pelatihan bobot layer membuat banyak neuron menerima input negatif terus-menerus, maka:

- Output neuron = 0
- Gradien turunan ReLU = 0
- Neuron tidak akan pernah di-update lagi → mati permanen (tidak belajar)

**b. Dampaknya terhadap CNN:**

- Banyak neuron mati di layer konvolusi awal → sinyal fitur penting dari gambar ikan tidak lolos ke layer selanjutnya.
- Aliran gradien selama backpropagation menjadi sangat lemah atau terputus.
- Ini menyebabkan stagnasi akurasi: model tidak belajar fitur baru, meskipun learning rate sudah optimal.

**c. Kapan Dying ReLU Terjadi?**

- Saat inisialisasi bobot besar/kecil secara ekstrem.
- Pada model dalam/deep CNN (seperti EfficientNet) yang tidak menggunakan strategi mitigasi.
- Bila dataset tidak seimbang (seperti pada klasifikasi spesies ikan), ReLU bisa lebih sering aktif di kelas mayoritas → minoritas makin sulit dipelajari.

**d. Solusi dan Mitigasi:**

1. Ganti ReLU dengan Aktivasi Lebih Stabil:
   Leaky ReLU: $f(x) = x$ jika $x>0$, $\alpha x$ jika $x<=0$ dengan $\alpha$ kecil (mis: 0.01)
   Tidak pernah 0 total → neuron tetap hidup.
   *Python:
   `tf.keras.layers.LeakyReLU(alpha=0.01)`
   Parametric ReLU (PReLU) atau ELU juga bisa digunakan.
2. Gunakan Batch Normalization
   - Menstabilkan distribusi input setiap layer.
   - Menghindari input ke ReLU yang terlalu negatif secara konsisten.
   - Catatan: Di analisis sebelumnya sudah disebutkan bahwa BatchNorm di tempat yang salah bisa memperburuk generalisasi, jadi penempatannya penting (ideal: sebelum/bersama ReLU).
3. Inisialisasi Bobot yang Tepat
   - Gunakan He Initialization yang cocok untuk ReLU:
     *Python
     `kernel_initializer='he_normal'`
   - Membantu menghindari bobot awal ekstrem yang membuat output layer negatif terus-menerus.

4. Monitoring dan Deteksi
    - Bisa dilakukan dengan melihat rasio aktivasi 0 di layer awal:
      *Python
      model.get_layer('conv2d').output.numpy() == 0
    - Jika proporsi output 0 sangat tinggi → tanda banyak neuron mati.

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!
   Jawab:
   **Analisis: Mengapa Class-Weighted Loss Gagal Meningkatkan Kinerja Spesies X?**
   Class-weighted loss (misalnya via class_weight di Keras) bertujuan memberi penalti lebih tinggi pada kelas minoritas untuk membantu pembelajaran seimbang.
   Namun, AUC-ROC Spesies X yang tetap rendah menunjukkan bahwa sekadar memberi bobot **tidak cukup**. Ada tiga penyebab potensial yang saling berkontribusi:
   a. Karakteristik Data Spesies X yang Ambigu atau Tidak Diskriminatif
      - Citra Spesies X mungkin tidak memiliki ciri visual unik (misalnya: pola sisik, bentuk sirip, warna latar).
      - Akibatnya, CNN kesulitan membedakan Spesies X dari spesies lain → probabilitas prediksi mendekati random.
      - Ini membuat AUC stagnan di sekitar 0.5–0.6.

      Mitigasi:
      - Lakukan visualisasi Grad-CAM untuk melihat apakah model bisa menemukan area yang relevan untuk Spesies X.
      - Tambahkan fitur metadata jika tersedia (ukuran, lokasi tangkapan, habitat).

   b. Distribusi Data yang Sangat Tidak Seimbang Secara Latent
      - Meskipun class_weight digunakan, bisa jadi jumlah variasi intra-class untuk Spesies X terlalu kecil.
      - Misal: hanya sedikit gambar Spesies X dengan sudut dan pencahayaan berbeda → CNN overfit ke subset sempit dari pola gambar Spesies X.

      Mitigasi:
      - Lakukan augmentasi data khusus untuk Spesies X (rotation, brightness, flipping).
      - Gunakan oversampling di level batch (seperti tf.data.experimental.sample_from_datasets).

   c. Arsitektur CNN Kurang Fleksibel untuk Kelas Minoritas

- CNN pretrained (seperti EfficientNetB0) belajar fitur dari dataset besar (ImageNet), yang tidak representatif untuk fine-grained klasifikasi seperti ikan.
- Model bisa belajar fitur general, tetapi gagal menangkap perbedaan halus antar Spesies.

Mitigasi:

- Unfreeze lebih banyak layer awal agar bisa belajar ulang fitur dasar yang sesuai untuk ikan.
- Tambahkan custom head layer yang lebih dalam dan eksplisit menangani perbedaan antar spesies.
- Coba loss function alternatif seperti Focal Loss, yang memfokuskan pembelajaran pada kelas yang sulit diprediksi.

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

   Jawab:

   Setelah meningkatkan kompleksitas arsitektur CNN untuk klasifikasi ikan, **akurasi training meningkat dari 90% → 98%**, tetapi **akurasi validasi justru turun dari 85% → 65%**.

   **AnalisisOverfitting**

   Fenomena ini merupakan contoh klasik overfitting, di mana model belajar terlalu detail atau noise pada data training, hingga kehilangan kemampuan untuk menggeneralisasi ke data yang tidak terlihat sebelumnya.

   Ciri khas overfitting:

   - Training loss terus menurun, validasi loss meningkat/turun lalu naik.
   - Akurasi training jauh lebih tinggi dibanding validasi.
   - AUC-ROC/Precision/Recall validasi menurun meski loss training kecil.

   Mengapa Penambahan Kompleksitas Justru Menurunkan Generalisasi?

   Menambah kapasitas model (jumlah layer, neuron, parameter) tidak selalu berdampak positif, karena:

   - Model menjadi terlalu fleksibel, mampu menghafal data training.
   - Tanpa regularisasi cukup, model gagal menangkap pola umum yang berlaku di luar data training.

- Model kompleks memerlukan lebih banyak data agar dapat dilatih dengan baik — jika tidak, hasilnya cenderung overfit.

**Kesalahan Desain Arsitektur yang Memicu Degradasi:**

a. Terlalu Banyak Layer atau Filter tanpa Regularisasi
   - Penambahan layer konvolusi (misal 6–8 layer) atau filter besar (512, 1024) secara agresif meningkatkan jumlah parameter.
   - Jika tidak diimbangi Dropout, L2 Regularization, atau BatchNorm, maka model belajar noise dan terjadi memorization, bukan generalisasi.

b. Tidak Menggunakan Dropout atau Terlalu Kecil
   - Tanpa dropout (misal 0.5 pada dense layer), model sulit menghindari overfitting.
   - Dropout membantu "mematikan neuron sementara" saat training untuk mencegah ketergantungan antar fitur.
   - Rekomendasi: Tambahkan Dropout(0.5) setelah fully connected layer dan pertimbangkan SpatialDropout2D setelah conv layer.

c. Fully Connected Layer Terlalu Besar
   - Misalnya penambahan Dense(1024) atau lebih tanpa seleksi fitur → terlalu banyak bobot yang harus dipelajari dari representasi konvolusi.
   - Tanpa global average pooling atau flattening yang tepat, model bisa overfit karena dimensi vektor fitur terlalu tinggi.