Analysis Kasus CNN

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?
   Fenomena vanishing gradient terjadi ketika back-propagate melewati banyak lapisan, terutama dengan aktivasi seperti sigmoid/tanh, lapisan terlalu dalam dan tidak adanya normalisasi data. Akibatnya, bobot lapisan awal jarang ter-update, model sulit belajar fitur rendah (low-level), lapisan awal yang tidak efektif dan waktu konvergensi meningkat drastic. Namun fenomena ini dapat di atasi dengan menggunakan:
   - aktivasi ReLU/LeakyReLU yang tidak membatasi gradien di satu sisi.
   - Inisialisasi He (He normal/He uniform) untuk ReLU

     ```
     layers.Conv2D(64, 3, activation='relu',
                   kernel_initializer='he_normal')
     ```

   Namun penggunaan Batch Normalization dapat memperburuk generalisasi, karena

   - Jika ditempatkan setelah aktivasi dan sebelum pooling/dropout yang sensitif, BN dapat mengurangi keacakan (noise) yang membantu regularisasi.
   - Data training–validation distribution mismatch: statistik batch (mean/var) di training beda jauh dengan val/test.

   Maka strategi alternatif yang dapat dilakukan adalah

   - Group Normalization atau Layer Normalization (tidak bergantung batch size).
   - Dropout setelah blok konvolusi untuk regularisasi.
   - Weight decay (L2 regularization) pada optimizer:

     ```
     tf.keras.optimizers.Adam(learning_rate=1e-3, decay=1e-5)
     ```

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?

   Penyebab potensial

   - Learning Rate Terlalu Kecil yang membuat gradien update sangat kecil dapat dikatakan model “jalan di tempat”
   - Inisialisasi bobot kurang tepat, jika semua bobot dimulai di nilai sama (atau terlalu kecil), gradien seragam → no symmetry breaking.

- Kesalahan model. Jika model terlalu sederhana maka kapasitas tak cukup, tidak bisa fitting → plateau. Dan jika terlalu kompleks maka saturasi neuron, exploding/vanishing gradient.

Penggunaan Cyclic Learning Rate dapat mengeluarkan dari local minima karena cara kerja dari CLR sendiri dapat LR dinaik-turunkan periodik antara batas rendah dan tinggi yang dapat mendorong model keluar dari local minima dangkal
contoh eksplorisasi :

```
from tensorflow.keras.callbacks import CyclicLR
clr = CyclicLR(base_lr=1e-5, max_lr=1e-3, step_size=2000., mode='triangular')
model.fit(..., callbacks=[clr])
```

Momentum pada optimizer SGD dapat memengaruhi konvergensi, prinsip kerja momentum di SGD mirip seperti di fisika yang dapat mengakumulasi kecepatan update gradien sehingga dapat memperlancar update. Jika momentum terlalu besar maka bisa overshoot sedangkan terlalu kecil mirip sgd biasa.

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 ReLU hanya output max(0, x). Jika bobot dan bias membuat input selalu negatif, neuron "mati" (output nol), gradien nol → tidak pernah update lagi.
   - Gangguan aliran gradien: Banyak neuron mati → kapasitas model berkurang drastis.

   Solusi :

   - LeakyReLU / Parametric ReLU:

     ```
     layers.Conv2D(64, 3),
     layers.LeakyReLU(alpha=0.1)
     ```

   - Inisialisasi He (perkuat nilai awal bobot positif dan negatif).
   - Thresholded ReLU atau ELU yang memiliki gradien non-nol walau input < 0.

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!
   Class weighted gagal dikarenakan :
   - Imbalance ekstrem: Jika contoh kelas X sangat sedikit, meski berat loss besar, model tetap kesulitan belajar fitur unik.
   - Overlap fitur: Gambar Spesies X mirip dengan kelas lain, model bingung membedakan.
   - Arsitektur kurang representatif: Depth/width terbatas untuk mengekstrak ciri halus spesies X.

Tiga Faktor penyebab :

- Kualitas data rendah, noise, sudut berbeda beda (fitur tidak konsisten)
- Augmentasi umum (flip, rotate) mungkin merusak cirikhas x
- Filter konvolusi awal terlalu kasar sehingga tidak menangkap detail pola sirip/warna.

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
   Fenomena overfitting terjadi dikarenakan kapasitas yang terlalu besar sehingga menyesuailan noise dan outlier, kapasitas yang besar jika memperngaruhi banyaknya parameter yang menyebabkan rentanya menangkap pola semu data baru di data training.
   Tiga kesalahan desain arsitektur :
   - Terlalu banyak unit dense / filter tanpa regulasi
   - Blok konvolusi yang berturut turut tanpa pooling
   - Tidak menggunakan Regularisasi (L1/L2 pada kernel, dropout, atau data augmentation kuat (mixup, CutMix)).