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?
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?
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!
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!
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 dan analisis:

1. Akurasi training 98% tetapi akurasi validasi 62% berarti terdapat indikasi overfitting.
   Fenomena vanishing gradient pada CNN dengan banyak lapisan, saat backpropagation, gradien mengecil drastis di layer awal karena:
   - Derivatif fungsi aktivasi seperti sigmoid/tanh < 1
   - Banyak perkalian matriks berturut-turut menyebabkan $\prod \partial L/\partial zi \rightarrow 0$

   Batch Normalization dapat memperburuk normalisasi karena:

   - Jika diterapkan terlalu dalam/tanpa regularisasi lain sehingga model overfit data training
   - Distribution shift internal bisa terlalu “dibekukan” sehingga mengganggu fitur fleksibel

   Walaupun dalam code, penambahan Batch Normalization menghasilkan hasil yang lebih bagus.

   Beberapa alternatifnya:

   - Skip Connections misal ResNet untuk membantu gradien mencapai layer awal
   - Dropout setelah dense layer untuk mengurangi overfitting
   - Data augmentation untuk meningkatkan generalisasi tanpa mengubah arsitektur
   - Menambahkan early stopping dan monitoring validation loss

2. Tiga kemungkinan penyebab:
   - Karena LR terlalu kecil
   - Inisialiasi bobot/weight buruk sehingga stuck di plateu
   - Modelnya terlalu kompleks sehingga sulit dioptimasi

   Alasan CLR dapat membantu:
   - CLR naik-turun secara periodik sehingga memungkinkan escape dari local minima dan saddle point
   - Bertindak seperti "shake-up" agar tidak stuck terlalu lama di loss tinggi

   Peran momentum pada optimizer SGD dapat menjaga arah gradien akumulatif dan gradien bisa tetap bergerak walau loss datar sesaat.

3. Dying Relu yaitu misal neuron output selalu 0 karena masuk ke zona negatif dari ReLU dan tidak pernah pulih, sehingga bobot tidak pernah diperbarui. Efeknya sejumlah neuron mati secara permanen dan stagnan.
   Solusinya bisa ganti ke LeakyReLU (sudah diaplikasikan di kode) atau ELU/Swish.

4. Walaupun saat training tidak mendapat satu spesies yang stagnan, namun misalkan ada maka penyebabnya bisa jadi karena:
   - Data gambar kurang variatif
   - Mirip seperti ikan lain
   - Model gagal untuk mempelajari

   Class-weighting tetap tidak bisa membantu model membedakan jika fitur tidak cukup informatif. Solusinya yaitu menambahkan modul attention juga evaluasi kualitas labeling & augmentasi spesifik untuk X.

5. Pada codingan terjadi pada Epoch 12 atau 14 namun pada epoch setelahnya akurasi validasi naik:

```
Epoch 11/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 303s 2s/step - accuracy: 0.8794 - loss: 0.3949 - val_accuracy: 0.9091 - val_loss: 0.3295
Epoch 12/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 287s 2s/step - accuracy: 0.9035 - loss: 0.3101 - val_accuracy: 0.7412 - val_loss: 1.1933
Epoch 13/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 306s 2s/step - accuracy: 0.9036 - loss: 0.3036 - val_accuracy: 0.9404 - val_loss: 0.2679
Epoch 14/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 340s 2s/step - accuracy: 0.9335 - loss: 0.2181 - val_accuracy: 0.8640 - val_loss: 0.4905
Epoch 15/15
138/138 ━━━━━━━━━━━━━━━━━━━━ 305s 2s/step - accuracy: 0.9370 - loss: 0.1936 - val_accuracy: 0.9451 - val_loss: 0.2427
```

   Kemungkinan overfitting terjadi karena model terlalu dalam/kompleks. Penambahan kapasitas model tidak selalu bagus karena tanpa regularisasi (dropout, L2, data augmentation), model terlalu fleksibel juga penambahan layer tidak berarti lebih pintar malahan bisa overparameterisasi.
   Tiga kesalahan desain arsitektur:
   1. Terlalu banyak layer konvolusi tanpa skip connection atau pooling
   2. Ukuran filter terlalu besar atau stride selalu kecil
   3. Tidak ada dropout/normalisasi di dense layer sehingga dense layer overfit