Pertanyaan Analisa CNN dari datasets Klasifikasi Ikan

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

Jawaban & Analisa:

1. Fenomena vanishing gradient pada CNN
   Vanishing Gradient terjadi Ketika gradient dari loss function menjadi sangat kecil saat mengalir ke belakang melalui banyak lapisan, khususnya lapisan awal. Akibatnya adalah:
   - Bobot pada lapisan awal hampir tidak diperbarui saat backpropagation.
   - CNN tidak belajar fitur dasar dengan baik.
   - Akurasi validasi buruk meskipun akurasi training tinggi

   Cara mitigasi vanishing gradient:

   - Gunakan fungsi aktivasi ReLU atau turunannya seperti LeakyReLU
   - Inisialisasi bobot yang tepat.
   - Tambahkan shortcut connection (ResNet).
   - Gunakan batch normalization di layer awal.

   Batch Normalization di lapisan ke-Y memperburuk generalisasi:

   - Apabila Batch Normalization diterapkan setelah lapisan konvolusi yang menangani fitur semantic tinggi, maka bisa mengganggu representasi yang sudah matang.

- Batch Normalization dapat menghilangkan distribusi alami fitur, terutama jika layer ke-Y berada terlalu dalam.
- Model menjadi sangat tergantung pada batch statistics selama training, sehingga tidak konsisten saat inferensi (validasi/test)

Strategi Alternatif:

- Gunakan Layer Normalization atau Group Normalization, khususnya jika batch size kecil atau distribusi batch sangat bervariasi.
- Terapkan Dropout atau SpatialDropout, maka akan membantu regularisasi dan mengurangi overfitting.
- Menggunakan Early Stopping dan data augmentation, maka akan menambah variasi data agar model tidak terlalu fit terhadap training set.
- Menggunakan arsitektur pretrained atau pre-activated seperti ResNet untuk tranfer learning.
- Gunakan Learing Rate Scheduler untuk mengatur dinamika pembelajaran lebih stabil dari awal ke akhir training.

2. Loss training stagnan di nilai tinggi setelah XXX (3-digit epoch) epoch.
   a. Learning Rate Tidak Tepat:
      - Terlalu kecil sehingga gradient descent sangat lambat dan membuat model butuh ratusan epoch tapi tetap setuck di loss tinggi.
      - Terlalu besar sehingga melewati titik minimum dan membuat loss bisa naik turun atau justru divergen.
   b. Inisialisasi Berat yang tidak sesuai:
      - Aktivasi bisa menghilang (vanishing) atau meledak (exploding).
      - Gradient menjadi nol atau sangat kecil sehingga menyebabkan stagnansi.
   c. Model terlalu kompleks atau terlalu sederhana:
      - Terlalu kompleks (overparameterized) sehingga model akan sulit dioptimasi, maka training akan stagnan tanpa regulasi
      - Terlalu sederhana (underfit) sehingga model tidak mampu merepresentasikan pola data, maka loss tetap tinggi meski epoch bertambah.

   Penggunaan Cyclic Learning Rate membantu model keluar dari local minima:

   - Variasi periodik learning rate menciptakan fluktuasi dalam pembaruan bobot.
   - Ketika learning rate meningkat sementara loss stagnan, model bisa melompat keluar dari local minima.
   - Saat learning rate menurun kembali, model bisa menemukan solusi yang lebih optimal.

   Momentum pada optimizer SGD memengaruhi konvergensi:

   - Menambahkan komponen dari gradien sebelumnya ke pembaruan saat ini.
   - Membantu menembus flat minima atau local minima.
   - Mengurangi osilasi di sepanjang arah gradien yang tidak konsisten.

3. Fenomena Dying ReLU

   Dying ReLU terjadi saat sejumlah besar neuron ReLU hanya menghasilkan nol secara permanen. Neuron tersebut mati dikarenakan tidak mengaktifkan apapun dan gradiennya nol sehingga tidak mengalami pembaruan selama training.

Bagaimana Dying ReLU Mengganggu Backpropagation:

- Gradien menghilang (Vanishing Gradient secara lokal):
  Karena output ReLU adalah 0, maka turunan fungsinya juga menjadi 0, sehingga tidak ada sinyal error yang mengalir ke neuron sebelumnya.
- Neuron Tidak Pernah Belajar Lagi:
  Begitu bobot membuat input menjadi negative maka neuron akan mati, sehingga tidak ada cara bagi neuron itu untuk bangkit kembali. Maka, lama-lama bagian jaringan menjadi tidak aktif secara permanen.

Untuk menghindari hal tersebut dapat dilakukan dengan cara mengganti ReLU dengan Leaky ReLU dan turunannya, menggunakan batch normalization untuk membantu menstabilkan distribusi input ke lapisan ReLU, dan perbaiki inisialisasi bobot dengan He Initialization.

4. Class-Weighted Loss Function Gagal:
   Analisis faktor:
   - AUC-ROC Spesies X (0.55) jauh lebih rendah dari kelas lain (>0.85) menunjukkan bahwa model kesulitan membedakan kelas Spesies X dari kelas lain

   Faktor penyebab potensial:

   a. Ketidakseimbangan Fitur
      - Ciri visual dari spesies X mungkin kurang distingtif jika dibandingkan dengan spesies lain. Hal tersebut membuat model kesulitan mempelajari perbedaan yang jelas antara kelas-kelas tersebut.
      - Solusi: melakukakn feature engineering yang lebih mendalam untuk menyaring dan menonjolkan ciri-ciri unik dari spesies X

   b. Variasi Intra-Kelas yang Tinggi:
      - Spesies X memiliki variasi penampilan yang lebih luas, sehingga gambar dari Spesies X bisa sangat bervariasi. Hal tersebut dapat menyebabkan kesulitan bagi model untuk menangkap representasi yang konsisten dan membedakan ciri khas Spesies X
      - Solusi: Menerapkan teknik augmentasi data yang berfokus pada Spesies X, seperti rotasi, perubahan skala, dan perubahan pencahayaan yang lebih beragam untuk mencakup lebih banyak variasi visual dari Spesies X.

   c. Data Berkualitas Rendah:
      - Kemungkinan ada noise atau kesalahan labeling pada dataset untuk kelas Spesies X. Labeling yang salah atau gambar yang buram/noisy dapat memperburuk performa model untuk kelas ini.
      - Solusi: Melakukan pembersihan dataset dengan memeriksa dan memperbaiki label serta menghilangkan gambar berkualitas rendah atau ambigus.

5. Penurunan Akurasi Validasi pada kompleksitas model:
   a. Fenomena Overfitting
      - Overfitting terjadi ketika model belajar sangat baik pada data pelatihan, bahkan pada detail dan noise yang tidak relevan, sehingga menyebabkan performa di data validasi menurun.

- Meskipun model dengan kapasitas lebih besar seperti lebih banyak lapisan atau neuron bisa menangkap lebih banyak fitur dari data, model yang terlalu kompleks cenderung menyesuaikan terlalu banyak detail pada data pelatihan, sehingga tidak mampu menggeneralisasi ke data yang baru.

b. Mengapa penambahan kapasitas model tidak selalu meningkatkan generalisasi:
   - Model yang lebih kompleks memiliki lebih banyak parameter dan kemampuan yang lebih besar untuk mempelajari noise dalam data pelatihan, sehingga tidak berlaku untuk data baru.
   - Overfitting dapat terjadi meskipun model lebih besar, dikarenakan model yang sangat kompleks akan mempelajari pola yang tidak penting atau kebetulan yang ada dalam data pelatihan, namun pola tersebut tidak ada pada data validasi.

c. Identifikasi 3 kesalahan desain arsitektur yang memicu degradasi performa:
   1. Arsitektur Terlalu Kompleks untuk Dataset yang Sederhana:
      - Penyebab:
        Menambah lapisan atau neuron pada model mungkin tidak selalu meningkatkan kinerja jika dataset tidak cukup besar atau rumit. Model yang terlalu kompleks justru menyebabkan overfitting, karena model mempelajari pola yang tidak relevan dengan data yang lebih luas.
      - Solusi:
        Menggunakan model yang lebih sederhana dengan regularisasi yang lebih baik seperti Dropout atau L2 regularization untuk mencegah model mengingat terlalu banyak detail yang tidak relevan.
   2. Tidak Menggunakan Regularisasi yang Cukup
      - Penyebab:
        Penambahan kapasitas model tanpa regularisasi yang memadai (seperti Dropout, Batch Normalization, atau L2 regularization) akan menyebabkan model "menghafal" data pelatihan tanpa berusaha mempelajari hubungan yang lebih umum.
      - Solusi:
        Implementasikan teknik regularisasi seperti Dropout, yang secara acak mematikan neuron selama pelatihan untuk memaksa model belajar representasi yang lebih robust, atau Batch Normalization untuk menstabilkan pelatihan dan mengurangi overfitting.
   3. Data Pelatihan Tidak Cukup atau Tidak Terdiversifikasi
      - Penyebab:
        Jika jumlah data pelatihan tidak cukup banyak atau tidak cukup bervariasi, model akan belajar pola yang sangat spesifik pada data pelatihan, yang mengarah pada overfitting.
      - Solusi:
        Perbanyak data pelatihan dengan teknik data augmentation (misalnya rotasi, flipping, perubahan warna) untuk meningkatkan keragaman dan membantu model menggeneralisasi lebih baik.