UTS Deep Learning – Fish Classification Model (Tensorflow)

Nama : Hasan Al Banna

NIM : 1103223142

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?**

   Hasil Analisa:

   Ketika arsitektur CNN memiliki X lapisan konvolusi dan akurasi training-nya tinggi banget (98%) tapi akurasi validasi-nya rendah (62%), itu tandanya modelnya overfitting. Salah satu penyebab yang mungkin terjadi, terutama jika modelnya dalam dan kompleks, biasa disebut dengan fenomena *vanishing gradient*.

   Vanishing gradient itu terjadi saat *backpropagation gradien* dari *loss* makin lama makin kecil pas ditelusurin ke lapisan awal. Akibatnya, lapisan-lapisan awal jadi jarang banget "belajar" karena update bobot yang kecil, bahkan bisa hampir nol. Ini bikin fitur dasar yang penting dari input (kayak tepi, bentuk, tekstur) tidak bisa dipelajari dengan baik.

   Cara mengatasi *vanishing gradient* ini antara lain:

   - Pakai aktivasi ReLU (atau variasinya kayak LeakyReLU) daripada sigmoid atau tanh, karena ReLU lebih stabil dan tidak gampang bikin gradien hilang.
   - Gunakan initialization yang tepat, kayak He initialization buat jaringan yang pakai ReLU.
   - Gunakan arsitektur yang udah terbukti stabil kayak ResNet yang punya *shortcut connection*, jadi gradien bisa "lompat" ke lapisan awal tanpa hilang.

   Nah, soal Batch Normalization: meskipun biasanya membantu stabilisasi training dan mempercepat konvergensi, kadang jika ditaruh setelah lapisan konvolusi ke-Y (apasedang di bagian akhir CNN yang udah spesifik fit ke data training), dia malah bisa bikin generalization ke data validasi makin buruk. Kenapa? Karena BN itu secara tidak langsung bikin distribusi fitur lebih homogen, padahal model kadang perlu fleksibilitas buat ngepasin variasi data real-world.

   Kalau BN justru bikin model makin overfit, alternatif lain yang bisa dicoba untuk menstabilkan pembelajaran:

   - Dropout: membantu mengurangi ketergantungan model ke neuron tertentu.
   - Data augmentation: agar model melihat variasi data yang lebih luas, tidak cuma fokus ke pattern yang sama terus.
   - Early stopping: berhentiin training sebelum model mulai overfit.
   - Atau, turunin kompleksitas model — mungkin X lapisan itu terlalu dalam buat dataset yang dipakai.

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?**
   Jawaban:

   Jika training CNN dari nol dan ternyata loss-nya stagnan di angka tinggi walaupun sudah lewat ratusan epoch (3 digit), berarti modelnya kesulitan untuk belajar. Ada beberapa kemungkinan penyebabnya, khususnya yang berhubungan dengan learning rate, inisialisasi bobot, dan kompleksitas model:

   - Learning rate terlalu kecil
     Jika learning rate-nya terlalu kecil, update bobotnya jadi pelan banget. Jadi meskipun udah dilatih ratusan epoch, loss-nya tidak turun-turun karena langkah perbaikannya terlalu kecil untuk keluar dari kondisi stuck atau dataran (plateau).
   - Inisialisasi bobot yang jelek
     Kalau bobot awalnya tidak di-set dengan baik (misal asal random, atau terlalu kecil/besar), jaringan bisa langsung masuk ke kondisi gradien yang kecil banget (vanishing) atau malah besar banget (exploding). Akibatnya, training bisa stuck dari awal dan loss-nya tidak bergerak signifikan.
   - Model terlalu kompleks (overparameterized)
     Arsitektur yang terlalu besar atau dalam membuat dataset yang kecil atau sederhana bisa bikin model susah untuk mencari pola yang optimal. Modelnya malah sibuk mengatur bobot yang tidak penting, akhirnya malah stuck di local minima yang jelek.

   Nah, di sinilah Cyclic Learning Rate (CLR) bisa bantu. Cyclic LR itu mengatur learning rate agar naik-turun dalam rentang tertentu, bukan turun terus kayak biasanya. Ini bisa membantu model dalam:

- Keluar dari local minima atau saddle point karena saat learning rate-nya naik, model bisa "lompat" dari titik-titik yang bikin stuck.
- Dapetin solusi yang lebih umum (generalizable), karena lonjakan learning rate sesekali membuat model eksplor parameter space lebih luas.

Lalu soal momentum di SGD, ini konsepnya seperti bola yang menggelinding di lereng:

- Momentum membantu model agar tetap "bergerak ke depan" walaupun gradien sedang kecil.
- Ini bikin konvergensi lebih cepat karena tidak terjebak bolak-balik di lembah sempit atau noise lokal.
- Selain itu, momentum bisa membantu stabilisasi arah gerakan update, jadi tidak terlalu zig-zag.

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!

Jawaban :

Akurasi yang mandek (tidak naik-naik) setelah 50 epoch padahal learning rate sudah diatur dengan benar, bisa jadi sedang ada masalah yang namanya dying ReLU.

Dying ReLU terjadi saat terlalu banyak neuron yang hasil output-nya selalu nol. Ini bisa terjadi karena:

- Input ke neuron tersebut negatif, dan karena ReLU = max(0, x), maka hasilnya nol.
- Setelah beberapa update, bobot membuat inputnya tetap negatif terus, akhirnya neuron itu "mati" — alias tidak pernah aktif lagi.

Bagaimana efeknya ke training?

Neuron yang sudah "mati" itu:

- Tidak punya output ≠ 0, jadi waktu backpropagation, gradiennya juga nol.
- Artinya, bobot neuron itu tidak akan pernah diperbarui lagi.
- Jika terlalu banyak neuron yang mati, CNN akan kehilangan kapasitas untuk belajar fitur penting dari gambar ikan tadi.
- Akhirnya aliran gradien ke lapisan sebelumnya juga ikut bermasalah, dan training bisa stuck, tidak berkembang.

Bagaimana Solusinya?

Ada beberapa cara untuk mengatasi atau mencegah dying ReLU:

- Ganti ReLU dengan Leaky ReLU, ELU, atau SELU
  Leaky ReLU tetap memberi output kecil waktu input-nya negatif, jadi neuron masih bisa dapet gradien dan tidak mati total.
- Perhatikan inisialisasi bobot
  Misalnya pakai He initialization yang lebih cocok buat ReLU dan turunannya, agar distribusi input antar layer tetap stabil.
- Pakai Batch Normalization
  Ini membantu menjaga nilai input neuron tetap dalam rentang yang lebih stabil, jadi tidak terlalu sering masuk ke area negatif ekstrim.

Jika menggunakan arsitektur yang sangat dalam atau datanya sulit dibedakan (misal ikan mirip-mirip bentuknya), masalah dying ReLU bisa makin kelihatan. Perlu hati-hati juga dalam milih aktivasi dan kombinasi layer.

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!**

Jawaban :

Jika pada grafik AUC-ROC, satu kelas misalnya Spesies X mentok di angka 0.55 (nyaris acak), padahal kelas-kelas lain sudah tembus 0.85 ke atas, berarti ada

masalah serius khusus di kelas itu. Meskipun sudah pakai class-weighted loss function, kok bisa tidak ngaruh ke performa Spesies X?

Kenapa class-weighted loss function bisa gagal?
Class-weight itu tujuannya untuk "menebus" kelas minoritas agar model lebih fokus ke sana. Tapi kadang tetap gagal meningkatkan performa jika:

- Model tidak bisa menemukan pola yang khas dari Spesies X, walaupun sudah dikasih bobot lebih tinggi.
- Fitur dari Spesies X terlalu mirip sama spesies lain, jadi model bingung membedakannya, alias low separability.
- Class-weight cuma bantu dari sisi loss, tapi tidak menambah informasi baru ke model.

Tiga penyebab potensial lainnya:

- Data Spesies X terlalu sedikit atau tidak representatif
  Jumlahnya bisa jadi cukup, tapi gambarnya tidak variatif (misal cuma dari satu sudut, kondisi cahaya mirip semua, atau kualitas buruk). Model jadi tidak dapet cukup "contoh unik" untuk belajar pola.
- Intra-class variance tinggi, inter-class variance rendah
  Spesies X mungkin punya bentuk, warna, atau ukuran yang sangat beragam, padahal antar spesies lain justru beda-beda jelas. Akibatnya model kesulitan membentuk boundary yang jelas buat Spesies X.
- Model architecture-nya tidak cukup peka atau terlalu general
  Bisa jadi arsitektur CNN yang terlalu "kasar" atau shallow, jadi tidak mampu menangkap detail-detail kecil yang penting untuk membedakan Spesies X dari lainnya. Atau kebalikannya, bisa juga terlalu kompleks dan malah overfit ke kelas lain yang lebih dominan.

Solusi tambahan yang bisa dicoba:

- Data augmentation khusus untuk Spesies X, seperti rotasi, crop, lighting variation, agar model belajar lebih banyak variasi dari kelas itu.
- Oversampling Spesies X secara sintetis (misal pakai SMOTE untuk citra) atau under-sampling kelas lain.
- Coba pendekatan lain kayak focal loss agar penalti lebih fokus ke prediksi yang susah.
- Jika ekstrim, bisa juga latih model binary khusus untuk mendeteksi Spesies X vs lainnya, lalu ensemble dengan model utama.

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 :

   Ketika meningkatkan kompleksitas model dalam CNN untuk klasifikasi ikan dan akurasi validasi turun dari 85% ke 65%, meskipun akurasi training tetap tinggi (98%), itu adalah fenomena overfitting. Overfitting terjadi ketika model belajar terlalu banyak dari data training hingga dia terlalu fit dengan data tersebut, bahkan mengingat

noise atau pola yang tidak penting, yang tidak generalizable ke data baru (seperti data validasi).

Kenapa penambahan kapasitas model tidak selalu meningkatkan generalisasi?
Penambahan kapasitas model, seperti menambah layer atau neuron, memang bisa meningkatkan kemampuan model dalam mempelajari fitur-fitur kompleks, tapi kadang ini juga membuat model terlalu rumit dan terlalu spesifik ke data training. Ini mengakibatkan:

- Model jadi terlalu peka terhadap noise di data training, dan kesulitan menangani data yang sedikit berbeda.
- Kapasitas model yang besar tanpa regularisasi atau teknik lain bisa membuat model overfit, jadi dia tidak bisa generalisasi dengan baik ke data baru.

Tiga kesalahan desain arsitektur yang dapat memicu degradasi performa:

- Model terlalu dalam atau overparameterized
  Jika model terlalu banyak lapisan atau neuron, dia bisa memiliki terlalu banyak kapasitas untuk mempelajari data. Ini membuat model terlalu baik di data training, tapi gagal menangkap pola yang lebih sederhana yang ada di data validasi atau test. Jadi, model overfit ke data training dan akurasi validasi turun.
- Kurangnya regularisasi (dropout, L2 regularization, dll.)
  Tanpa teknik regularisasi yang tepat, model bisa belajar terlalu baik di data training, bahkan dengan fitur-fitur yang sebenarnya tidak relevan atau noise. Misalnya, dropout bisa membantu model supaya tidak bergantung terlalu banyak pada fitur tertentu, dan L2 regularization bisa mengurangi bobot model yang terlalu besar. Tanpa ini, model cenderung terlalu fit.
- Data yang tidak cukup bervariasi atau terlalu kecil
  Jika dataset-nya terlalu kecil atau terlalu mirip satu sama lain, model akan kesulitan belajar pola yang benar-benar mewakili variasi data di dunia nyata. Misalnya, ketika dataset cuma punya gambar ikan dengan satu sudut pandang atau kondisi cahaya yang seragam, model bisa belajar untuk mengenali pola yang hanya ada di dataset itu dan gagal mengenali data yang lebih variatif.

Solusi untuk mengatasi overfitting:

- Gunakan teknik regularisasi seperti dropout, L2 regularization, atau batch normalization.
- Terapkan data augmentation agar model bisa belajar dari variasi data yang lebih luas dan tidak cuma bergantung ke data training.
- Coba early stopping untuk menghentikan training sebelum model mulai overfit.
- Coba mengurangi kompleksitas model (kurangi jumlah layer atau neuron) supaya model lebih sederhana dan lebih cepat generalisasi.