TECHNICAL REPORT

Muhammad Daffa Satria 1103204018

I. Pendahuluan

Deep learning adalah cabang dari pembelajaran mesin (machine learning) yang bertujuan untuk melatih model yang dapat mempelajari representasi yang mendalam (hierarchical) dari data. Model deep learning yang kompleks, seperti jaringan saraf tiruan (neural networks), telah terbukti sangat sukses dalam berbagai tugas seperti pengenalan gambar, pemrosesan bahasa alami, dan prediksi.

PyTorch adalah sebuah framework deep learning yang populer dan kuat yang dikembangkan oleh Facebook AI Research. PyTorch menawarkan fleksibilitas dan ekspresivitas tinggi dalam mengembangkan model deep learning. Hal ini membuatnya menjadi salah satu pilihan utama bagi para peneliti dan praktisi yang ingin mengimplementasikan dan menguji konsep-konsep baru dalam deep learning.

PyTorch menyediakan berbagai fitur yang mempermudah implementasi model deep learning, seperti tensor operations yang efisien, kemampuan autograd untuk menghitung gradien secara otomatis, dan alat-alat untuk memuat, menyimpan, dan menganalisis model. PyTorch juga menyediakan API yang intuitif dan dokumentasi yang kaya, yang memudahkan pengguna untuk memahami dan mengimplementasikan berbagai konsep dalam deep learning.

Laporan teknis ini bertujuan untuk memberikan gambaran tentang konsep dan fungsionalitas utama mengenai dasar-dasar tensor, autograd, backpropagation, gradient descent, alur latihan, regresi linear, regresi logistik, dataset dan Dataloader, transformasi dataset, softmax dan crossentropy, fungsi aktivasi, jaringan feed forward, jaringan saraf konvolusi (CNN), transfer learning, tensorboard, dan menyimpan/memuat model.

II. Dasar-dasar Tensor

Tensor adalah struktur data utama dalam komputasi menggunakan library PyTorch. Secara sederhana, tensor dapat dianggap sebagai array multidimensi yang mirip dengan konsep matriks atau vektor dalam matematika. Namun, tensor memiliki fleksibilitas yang lebih tinggi karena dapat memiliki lebih dari dua dimensi.

Dalam konteks PyTorch, tensor adalah objek yang dapat menyimpan dan memanipulasi data numerik secara efisien. Tensor dapat berisi bilangan bulat, bilangan riil, atau bahkan bilangan kompleks. Selain itu, tensor juga mendukung operasi aljabar linear seperti penjumlahan, pengurangan, perkalian, dan lainnya. Tensor juga merupakan struktur data yang kompatibel dengan operasi komputasi paralel yang dilakukan pada perangkat keras seperti GPU, yang memungkinkan pemrosesan yang lebih cepat untuk tugas-tugas yang memerlukan komputasi yang intensif.

Tensor dalam PyTorch memiliki beberapa atribut penting, seperti bentuk (shape) yang menunjukkan ukuran dari setiap dimensi tensor, tipe data (data type) yang menentukan jenis bilangan yang disimpan dalam tensor, dan perangkat (device) yang menunjukkan apakah tensor disimpan dan diolah pada CPU atau GPU.

Tensor juga memungkinkan pelacakan gradien, yang merupakan fitur penting dalam komputasi numerik dan pembelajaran mesin. Dengan pelacakan gradien, tensor dapat digunakan dalam proses optimisasi seperti penyesuaian parameter dalam algoritma pembelajaran mesin seperti backpropagation.

Dalam praktiknya, tensor digunakan secara luas dalam berbagai tugas seperti pengolahan citra, pengenalan suara, pemrosesan bahasa alami, dan pembelajaran mesin secara umum. PyTorch menyediakan fungsi dan metode yang kuat untuk membuat, menginisialisasi, memanipulasi, dan melakukan operasi matematika pada tensor, sehingga memudahkan pengguna untuk mengembangkan dan menerapkan model dan algoritma yang kompleks.

Selain membuat tensor dari awal, pada bagian ini juga menunjukkan cara membuat tensor dari data yang sudah ada menggunakan fungsi torch.tensor(). Bagian ini juga menjelaskan tentang argumen requires_grad yang digunakan untuk menandai tensor sebagai variabel yang perlu dihitung gradiennya saat melaksanakan langkah-langkah optimisasi.

Tensor juga dapat melakukan operasi seperti penjumlahan, pengurangan, perkalian, dan pembagian elemen-wise. Pada bagian ini juga menunjukkan bagaimana melakukan slicing pada tensor untuk mengakses subset elemennya.

III. Autograd

Autograd (automatic differentiation) adalah mekanisme dalam PyTorch yang memungkinkan perhitungan gradien otomatis. Gradien adalah turunan dari suatu fungsi terhadap parameter atau variabel tertentu. Dalam konteks pembelajaran mesin, gradien sangat penting dalam proses optimisasi, di mana kita mencoba menemukan nilai parameter yang optimal untuk model kita.

Autograd bekerja dengan cara merekam grafik komputasi yang terdiri dari operasi-operasi matematika yang diterapkan pada tensor. Setiap tensor memiliki atribut requires_grad yang dapat diatur sebagai True atau False. Ketika requires_grad diatur sebagai True, tensor tersebut akan mengikuti dan melacak setiap operasi matematika yang diterapkan padanya.

Ketika kita melakukan perhitungan pada tensor yang memiliki requires_grad=True, PyTorch secara otomatis membangun grafik komputasi (komputasi grafis) yang mencatat setiap operasi yang dilakukan dan membangun alur perhitungan. Dengan menggunakan algoritma backpropagation, PyTorch kemudian dapat menghitung gradien dari hasil akhir terhadap tensor-tensor yang memiliki requires_grad=True.

Dengan adanya autograd, kita dapat dengan mudah menghitung gradien dan melakukan optimisasi model secara otomatis. Autograd sangat berguna dalam pembelajaran mesin, terutama ketika kita menggunakan algoritma pembelajaran berbasis gradien seperti backpropagation pada jaringan saraf tiruan.

Dalam praktiknya, kita dapat mengaktifkan autograd dengan mengatur requires_grad=True pada tensor yang ingin dilacak gradiennya. Kemudian, ketika kita ingin menghitung gradien, kita dapat memanggil metode backward() pada tensor hasil akhir, dan gradien akan dihitung secara otomatis untuk setiap tensor yang terlibat dalam perhitungan.

Secara keseluruhan, autograd memungkinkan kita untuk dengan mudah menghitung gradien, yang merupakan komponen penting dalam optimisasi model dan pembelajaran mesin secara umum.

IV. Backpropagation

Backpropagation adalah algoritma yang digunakan untuk menghitung gradien (turunan parsial) dari fungsi kerugian (loss function) terhadap parameter-parameter yang ada dalam model pembelajaran mesin, terutama pada jaringan saraf tiruan (neural networks).

Dalam proses pelatihan model, tujuan kita adalah untuk menemukan nilai parameter yang optimal yang dapat menghasilkan prediksi yang akurat. Untuk mencapai hal ini, kita meminimalkan fungsi kerugian yang mengukur kesalahan antara prediksi model dengan nilai yang sebenarnya. Dalam jaringan saraf tiruan, fungsi kerugian ini terhubung dengan parameter-parameter model melalui serangkaian operasi matematika yang kompleks.

Backpropagation bekerja dengan memanfaatkan aturan rantai (chain rule) dalam kalkulus untuk menghitung gradien fungsi kerugian terhadap setiap parameter dalam model. Proses ini terjadi secara mundur, yaitu dimulai dari nilai kesalahan pada output dan kemudian menghitung gradien mundur melalui setiap lapisan jaringan hingga mencapai parameter-parameter awal.

Secara umum, proses backpropagation terdiri dari beberapa tahap, yaitu:

Feedforward: Input diteruskan melalui jaringan dan menghasilkan prediksi output.
Perhitungan kesalahan (loss calculation): Membandingkan prediksi model dengan nilai yang sebenarnya dan mengukur kesalahan menggunakan fungsi kerugian.
Backward pass: Gradien kesalahan dihitung mundur melalui setiap lapisan jaringan menggunakan aturan rantai (chain rule) kalkulus. Gradien ini menggambarkan seberapa sensitif nilai kesalahan terhadap perubahan parameter.
Perbarui parameter: Setelah mendapatkan gradien, kita dapat menggunakan algoritma optimisasi seperti gradien turun (gradient descent) untuk memperbarui nilai parameter agar mengurangi kesalahan pada iterasi berikutnya.
Iterasi: Proses diulangi dengan menggunakan data pelatihan yang berbeda dan parameter yang diperbarui untuk meningkatkan akurasi model.

Backpropagation merupakan inti dari pelatihan jaringan saraf tiruan dan memungkinkan model untuk mempelajari representasi yang lebih baik dari data pelatihan seiring dengan memperbarui parameter-model.

V. Gradient Descent

Gradient Descent (desent/penurunan gradien) adalah algoritma optimisasi yang digunakan untuk menemukan nilai minimum dari suatu fungsi. Tujuan utama Gradient Descent adalah untuk meminimalkan fungsi biaya (cost function) dalam konteks pembelajaran mesin.

Algoritma ini bekerja dengan mengiterasi nilai-nilai parameter model secara bertahap, berdasarkan gradien (turunan parsial) dari fungsi biaya terhadap parameter tersebut. Secara umum, langkah-langkah dalam Gradient Descent adalah sebagai berikut:

Inisialisasi parameter: Memilih nilai awal untuk parameter yang akan dioptimasi.
Perhitungan gradien: Menghitung gradien (turunan parsial) dari fungsi biaya terhadap setiap parameter. Gradien ini mengindikasikan arah dan tingkat peningkatan fungsi

biaya.
Pembaruan parameter: Menggunakan gradien yang dihitung, parameter diperbarui dengan mengurangi sejumlah kecil (learning rate) dikali gradien. Hal ini dilakukan untuk menggerakkan nilai parameter ke arah yang mengurangi fungsi biaya.
Iterasi: Langkah-langkah 2 dan 3 diulangi secara berulang hingga mencapai kondisi berhenti tertentu, seperti jumlah iterasi maksimum atau perubahan fungsi biaya yang tidak signifikan.
Pada setiap iterasi, Gradient Descent mengubah nilai parameter dengan langkah yang lebih kecil jika gradien tersebut masih menunjukkan penurunan fungsi biaya. Tujuan akhirnya adalah mencapai titik di mana gradien mendekati atau sama dengan nol, yang menunjukkan minimum lokal atau global dari fungsi biaya.

Terdapat beberapa variasi dari Gradient Descent, seperti Batch Gradient Descent, Stochastic Gradient Descent, dan Mini-Batch Gradient Descent. Perbedaannya terletak pada bagaimana data pelatihan digunakan untuk menghitung gradien dan memperbarui parameter. Gradient Descent (desent/penurunan gradien) adalah algoritma optimisasi yang digunakan untuk menemukan nilai minimum dari suatu fungsi. Tujuan utama Gradient Descent adalah untuk meminimalkan fungsi biaya (cost function) dalam konteks pembelajaran mesin.

Algoritma ini bekerja dengan mengiterasi nilai-nilai parameter model secara bertahap, berdasarkan gradien (turunan parsial) dari fungsi biaya terhadap parameter tersebut. Secara umum, langkah-langkah dalam Gradient Descent adalah sebagai berikut:

Inisialisasi parameter: Memilih nilai awal untuk parameter yang akan dioptimasi.
Perhitungan gradien: Menghitung gradien (turunan parsial) dari fungsi biaya terhadap setiap parameter. Gradien ini mengindikasikan arah dan tingkat peningkatan fungsi biaya.
Pembaruan parameter: Menggunakan gradien yang dihitung, parameter diperbarui dengan mengurangi sejumlah kecil (learning rate) dikali gradien. Hal ini dilakukan untuk menggerakkan nilai parameter ke arah yang mengurangi fungsi biaya.
Iterasi: Langkah-langkah 2 dan 3 diulangi secara berulang hingga mencapai kondisi berhenti tertentu, seperti jumlah iterasi maksimum atau perubahan fungsi biaya yang tidak signifikan.
Pada setiap iterasi, Gradient Descent mengubah nilai parameter dengan langkah yang lebih kecil jika gradien tersebut masih menunjukkan penurunan fungsi biaya. Tujuan akhirnya adalah mencapai titik di mana gradien mendekati atau sama dengan nol, yang menunjukkan minimum lokal atau global dari fungsi biaya.

Terdapat beberapa variasi dari Gradient Descent, seperti Batch Gradient Descent, Stochastic Gradient Descent, dan Mini-Batch Gradient Descent. Perbedaannya terletak pada bagaimana data pelatihan digunakan untuk menghitung gradien dan memperbarui parameter.

VI. Training Pipeline

Training Pipeline (pipa pelatihan) adalah langkah-langkah yang diperlukan dalam proses pelatihan (training) model dalam pembelajaran mesin. Ini mencakup serangkaian tindakan yang dilakukan untuk mempersiapkan data, membangun model, mengoptimalkan parameter, dan mengevaluasi kinerja model. Berikut adalah penjelasan singkat mengenai langkah-langkah dalam Training Pipeline:

Preprocessing Data: Langkah pertama dalam Training Pipeline adalah melakukan preprocessing data. Ini melibatkan pembersihan, transformasi, dan normalisasi data

agar sesuai dengan format yang dibutuhkan oleh model. Contohnya termasuk mengisi nilai yang hilang, mengubah data kategorikal menjadi numerikal, dan membagi data menjadi set pelatihan dan pengujian.

Memilih Model: Setelah data diproses, langkah berikutnya adalah memilih model yang sesuai untuk tugas pembelajaran mesin yang diberikan. Ini dapat berupa model regresi, model klasifikasi, atau model lainnya yang sesuai dengan masalah yang akan diselesaikan.

Membangun Model: Setelah model dipilih, langkah selanjutnya adalah membangun model itu sendiri. Ini melibatkan menentukan struktur model, seperti jumlah lapisan dan jumlah unit di setiap lapisan, serta fungsi aktivasi yang digunakan. Model ini biasanya diimplementasikan menggunakan kerangka kerja atau pustaka pembelajaran mesin, seperti TensorFlow atau PyTorch.

Mengoptimalkan Parameter: Setelah model dibangun, parameter model perlu dioptimalkan untuk meminimalkan fungsi biaya atau maksimalkan kinerja model. Ini dilakukan melalui algoritma optimisasi, seperti Gradient Descent, yang mengubah nilai parameter secara iteratif berdasarkan gradien fungsi biaya terhadap parameter.

Melakukan Pelatihan: Dalam langkah ini, model dilatih menggunakan data pelatihan. Ini melibatkan memberikan data pelatihan ke model, melakukan perhitungan maju (forward propagation) untuk mendapatkan prediksi, menghitung fungsi biaya antara prediksi dan label yang sebenarnya, dan melakukan perhitungan mundur (backpropagation) untuk menghitung gradien dan memperbarui parameter.

Evaluasi dan Validasi: Setelah model dilatih, langkah selanjutnya adalah mengevaluasi kinerjanya. Ini melibatkan pengujian model pada data pengujian yang terpisah untuk mengukur akurasi, presisi, keakuratan, atau metrik evaluasi lainnya yang relevan dengan tugas yang diberikan. Evaluasi ini membantu memahami seberapa baik model bekerja dan apakah perlu dilakukan penyesuaian tambahan.

Penyempurnaan Model: Jika hasil evaluasi tidak memuaskan, langkah ini melibatkan penyesuaian model, seperti penyesuaian hiperparameter, perubahan arsitektur model, atau perbaikan data pelatihan. Proses ini diulang kembali dengan membangun ulang model, melatihnya, dan mengevaluasinya hingga kinerja yang diinginkan tercapai.

Training Pipeline adalah rangkaian langkah yang penting dalam pengembangan model pembelajaran mesin

VII. Linear Regression

Linear Regression (Regresi Linear) adalah salah satu metode yang digunakan dalam analisis statistik untuk memodelkan hubungan linier antara variabel independen (variabel penjelas) dan variabel dependen (variabel yang ingin diprediksi). Tujuan dari Regresi Linear adalah untuk mengidentifikasi dan memahami pola atau hubungan linier antara variabel-variabel tersebut.

Konsep dasar dari Regresi Linear adalah memodelkan hubungan antara variabel dependen (Y) dan satu atau lebih variabel independen (X) dengan menggunakan persamaan garis lurus. Persamaan umum untuk Regresi Linear adalah:

$Y = b0 + b1*X$

Di mana Y adalah variabel dependen, X adalah variabel independen, b0 adalah intercept (titik perpotongan dengan sumbu Y) dan b1 adalah koefisien (gradien atau kemiringan garis regresi).

Proses Regresi Linear melibatkan:

Pengumpulan Data: Pertama, data yang relevan untuk variabel dependen dan variabel independen dikumpulkan. Data ini dapat berupa pasangan nilai yang telah diamati atau diukur dari kedua variabel.

Analisis Data: Data yang dikumpulkan kemudian dianalisis untuk mengidentifikasi pola atau hubungan linier yang ada antara variabel dependen dan variabel independen. Grafik sebaran (scatter plot) sering digunakan untuk memvisualisasikan hubungan antara variabel-variabel tersebut.

Estimasi Parameter: Selanjutnya, menggunakan metode kuadrat terkecil (least squares method), parameter (koefisien dan intercept) dalam persamaan Regresi Linear diestimasi. Tujuan utamanya adalah untuk menemukan garis regresi yang paling dekat dengan titik-titik data yang diamati.

Evaluasi Model: Setelah parameter diperoleh, model Regresi Linear dievaluasi untuk menentukan seberapa baik model tersebut cocok dengan data yang diamati. Metrik evaluasi seperti R-squared, mean squared error (MSE), atau root mean squared error (RMSE) digunakan untuk mengukur kualitas dan akurasi model.

Prediksi: Dengan menggunakan persamaan Regresi Linear yang diperoleh, model dapat digunakan untuk membuat prediksi nilai variabel dependen berdasarkan nilai variabel independen yang baru atau tidak diamati sebelumnya.

Regresi Linear cocok untuk masalah di mana hubungan antara variabel independen dan variabel dependen diasumsikan linier. Namun, jika hubungan tersebut kompleks atau tidak linier, metode lain seperti Regresi Non-linear atau metode pembelajaran mesin yang lebih kompleks mungkin lebih cocok.

VIII. Logistic Regression

Logistic Regression (Regresi Logistik) adalah metode statistik yang digunakan untuk memodelkan hubungan antara variabel independen dan variabel dependen yang berbentuk biner (dua kategori) atau berurutan (ordinal). Tujuan dari Regresi Logistik adalah untuk memprediksi probabilitas atau kemungkinan kejadian suatu peristiwa berdasarkan nilai-nilai variabel independen.

Regresi Logistik menggunakan fungsi logit (log-odds) untuk memodelkan hubungan antara variabel independen dan variabel dependen. Fungsi logit mengambil bentuk sebagai berikut:

$$logit(p) = ln(p / (1 - p))$$

Di mana p adalah probabilitas kejadian suatu peristiwa dan logit(p) adalah log-odds dari peristiwa tersebut. Persamaan Regresi Logistik dapat ditulis sebagai:

$$logit(p) = b0 + b1X1 + b2X2 + ... + bn*Xn$$

Di mana X1, X2, ..., Xn adalah variabel independen, b0 adalah intercept, dan b1, b2, ..., bn adalah koefisien (pengaruh) yang sesuai dengan setiap variabel independen.

Proses Regresi Logistik melibatkan:

Pengumpulan Data: Data yang relevan tentang variabel independen dan variabel dependen dikumpulkan. Data ini berisi pasangan nilai-nilai yang menggambarkan kejadian atau tidaknya suatu peristiwa, serta nilai-nilai variabel independen yang berkaitan.

Analisis Data: Data yang dikumpulkan dianalisis untuk mengidentifikasi pola dan hubungan antara variabel independen dan variabel dependen. Grafik dan tabel ringkasan statistik digunakan untuk memahami karakteristik data.

Estimasi Parameter: Parameter dalam persamaan Regresi Logistik (koefisien dan intercept) diestimasi menggunakan metode Maximum Likelihood Estimation (MLE) atau metode optimasi numerik lainnya. Tujuan utamanya adalah untuk menemukan nilai-nilai parameter yang paling cocok dengan data yang diamati.

Evaluasi Model: Setelah parameter diperoleh, model Regresi Logistik dievaluasi untuk menentukan seberapa baik model tersebut cocok dengan data yang diamati. Metrik evaluasi seperti Akurasi, Presisi, Recall, dan F1-Score digunakan untuk mengukur kualitas dan kinerja model.

Prediksi: Model Regresi Logistik yang diperoleh dapat digunakan untuk memprediksi probabilitas atau kategori kejadian suatu peristiwa berdasarkan nilai-nilai variabel independen yang baru atau tidak diamati sebelumnya.

Regresi Logistik banyak digunakan dalam berbagai bidang seperti ilmu sosial, kedokteran, keuangan, dan ilmu data. Metode ini cocok untuk masalah klasifikasi biner, seperti prediksi kejadian atau tidaknya suatu peristiwa berdasarkan variabel independen yang diberikan. Namun, jika klasifikasi memiliki lebih dari dua kategori, metode lain seperti Regresi Logistik

Multinomial atau Metode Pembelajaran Mesin yang lebih kompleks mungkin lebih sesuai.

IX. Dataset dan Dataloader

Dataset dan DataLoader adalah dua komponen penting dalam pemrosesan data dalam deep learning.

Dataset merujuk pada kumpulan data yang digunakan untuk melatih dan menguji model machine learning. Dataset dapat terdiri dari input (fitur) dan target (label) yang sesuai. Tujuan utama dari Dataset adalah menyediakan akses yang mudah ke data dan memungkinkan manipulasi yang efisien.

Dalam praktiknya, Dataset sering kali diimplementasikan sebagai kelas yang menggambarkan struktur data yang berisi data yang akan digunakan dalam pelatihan atau pengujian model. Dataset dapat menyediakan metode seperti getitem untuk mengakses sampel data tertentu dan len untuk mengembalikan jumlah total sampel dalam dataset.

Dataloader, di sisi lain, adalah komponen yang bertanggung jawab untuk memuat dan mengelola batch data dari Dataset. Dataloader memungkinkan pemrosesan data dalam batch, yang berguna untuk melatih model dengan efisiensi yang lebih tinggi. Dataloader secara otomatis membagi Dataset menjadi batch berukuran tertentu, melakukan pengacakan (shuffle) jika diinginkan, dan dapat memanfaatkan multiprocessing untuk mempercepat proses pemuatan data.

Dengan menggunakan Dataloader, pengguna dapat mengiterasi melalui batch data dengan mudah dan efisien dalam proses pelatihan dan pengujian model. Dataloader juga menyediakan fleksibilitas dalam mengatur ukuran batch, pengacakan data, dan penggunaan multi-pemrosesan (multiprocessing) untuk mempercepat pengolahan data.

Secara keseluruhan, Dataset dan DataLoader adalah komponen penting dalam memproses data dalam deep learning. Dataset memberikan akses yang mudah ke data, sedangkan Dataloader memungkinkan pemrosesan data dalam batch yang efisien. Kombinasi dari keduanya memungkinkan proses pelatihan dan pengujian model yang lebih efektif dan cepat.

X. Dataset Transforms

Transformasi dataset adalah proses mengubah atau memanipulasi data dalam dataset sebelum digunakan untuk melatih atau menguji model. Transformasi ini dapat melibatkan berbagai operasi seperti pemotongan (cropping), perubahan ukuran (resizing), pemutaran (rotation), normalisasi, augmentasi data, dan banyak lagi.

Transformasi dataset penting dalam deep learning karena dapat membantu dalam mempersiapkan data dengan cara yang sesuai untuk memenuhi kebutuhan model. Beberapa alasan umum untuk menggunakan transformasi dataset adalah:

Prapemrosesan data: Transformasi dataset dapat digunakan untuk melakukan prapemrosesan data seperti normalisasi, standarisasi, atau pengurangan dimensi. Ini membantu dalam mengoptimalkan kualitas data dan mempercepat proses pelatihan model.

Augmentasi data: Transformasi dataset dapat digunakan untuk menghasilkan variasi tambahan dalam data pelatihan dengan melakukan operasi seperti pemutaran, pergeseran, atau flipping. Ini membantu dalam meningkatkan keragaman data dan

mencegah overfitting pada model.

Penyesuaian format: Transformasi dataset dapat digunakan untuk mengubah format data ke format yang kompatibel dengan model yang akan digunakan. Misalnya, mengonversi gambar menjadi tensor atau mengubah label menjadi format yang sesuai.

Transformasi dataset seringkali diterapkan menggunakan fungsi atau kelas transformasi yang tersedia dalam pustaka seperti PyTorch atau TensorFlow. Pustaka-pustaka ini menyediakan berbagai transformasi yang siap pakai untuk mengubah data dalam dataset.

Dalam implementasinya, transformasi dataset dapat diterapkan secara langsung pada setiap sampel dataset menggunakan metode getitem di dalam kelas dataset. Selain itu, transformasi juga dapat diterapkan menggunakan komponen Dataloader dengan menyediakan parameter transform saat memuat dataset ke dalam Dataloader.

Secara keseluruhan, transformasi dataset adalah proses penting dalam pemrosesan data dalam deep learning. Ini membantu dalam mempersiapkan data dengan cara yang sesuai untuk pelatihan dan pengujian model. Transformasi dataset memungkinkan prapemrosesan data, augmentasi data, dan penyesuaian format data secara efisien.

XI. Softmax dan Crossentropy

Softmax dan Crossentropy adalah dua konsep penting dalam pemodelan klasifikasi multikelas.

Softmax adalah fungsi aktivasi yang digunakan untuk menghasilkan probabilitas kelas pada output layer dari model klasifikasi multikelas. Fungsi ini mengambil input berupa nilai numerik dan menghasilkan distribusi probabilitas untuk setiap kelas yang mungkin. Softmax memastikan bahwa probabilitas semua kelas yang mungkin akan berjumlah satu. Dengan menggunakan softmax, kita dapat menginterpretasikan output model sebagai probabilitas kelas yang dihasilkan.

Crossentropy, atau lebih tepatnya cross-entropy loss, adalah fungsi loss yang umum digunakan dalam klasifikasi multikelas. Fungsi ini mengukur perbedaan antara distribusi probabilitas yang dihasilkan oleh model (melalui softmax) dengan distribusi probabilitas yang sebenarnya dari data. Crossentropy berusaha untuk meminimalkan perbedaan ini dengan mengoptimalkan parameter-model melalui proses pelatihan.

Dalam konteks klasifikasi multikelas, crossentropy loss sering digunakan bersama dengan softmax. Dalam langkah-langkah pelatihan, output dari model melewati fungsi softmax untuk menghasilkan probabilitas kelas, kemudian probabilitas ini digunakan bersama dengan label yang sebenarnya untuk menghitung nilai loss dengan menggunakan crossentropy. Tujuan pelatihan adalah untuk menemukan parameter-model yang dapat meminimalkan nilai crossentropy loss, sehingga model dapat memberikan probabilitas yang lebih baik dalam mengklasifikasikan data yang belum pernah dilihat sebelumnya.

Dengan demikian, softmax dan crossentropy saling terkait dalam klasifikasi multikelas. Softmax digunakan untuk menghasilkan probabilitas kelas, sementara crossentropy digunakan untuk mengukur perbedaan antara probabilitas yang dihasilkan dengan probabilitas yang sebenarnya.

XII. Activation Functions

Activation functions, atau fungsi aktivasi, adalah fungsi matematika yang diterapkan pada output suatu neuron dalam jaringan saraf tiruan (neural network). Fungsi ini bertujuan untuk memperkenalkan non-linearitas pada model sehingga model dapat mempelajari hubungan yang kompleks antara input dan output.

Beberapa contoh fungsi aktivasi yang umum digunakan dalam neural network adalah sebagai berikut:

Sigmoid: Fungsi sigmoid memiliki bentuk S yang melimitkan output ke rentang 0 hingga 1. Fungsi ini digunakan terutama dalam kasus klasifikasi biner atau ketika ingin menghasilkan probabilitas sebagai output.

ReLU (Rectified Linear Unit): Fungsi ReLU memungkinkan output positif melewati tanpa perubahan, sementara input negatif diubah menjadi nol. Fungsi ini cenderung memberikan hasil yang lebih cepat dan efisien dalam pelatihan model.

Tanh (Hyperbolic Tangent): Fungsi tanh memiliki bentuk mirip dengan sigmoid, tetapi menghasilkan output di rentang -1 hingga 1. Fungsi ini juga sering digunakan dalam klasifikasi biner atau ketika ingin menghasilkan output yang memiliki skala dari -1 hingga 1.

Leaky ReLU: Leaky ReLU adalah variasi dari fungsi ReLU yang memungkinkan input negatif memiliki nilai kecil yang tidak nol, bukan hanya menjadi nol. Hal ini membantu mengatasi masalah "dying ReLU" di mana neuron dapat berhenti belajar karena outputnya selalu nol.

Softmax: Fungsi softmax umumnya digunakan pada output layer dari model klasifikasi multikelas. Fungsi ini mengonversi input menjadi probabilitas distribusi yang sesuai dengan jumlah kelas yang mungkin. Output dari softmax akan berjumlah satu, sehingga dapat diinterpretasikan sebagai probabilitas kelas yang dihasilkan oleh model.

Pemilihan fungsi aktivasi yang tepat tergantung pada karakteristik masalah yang dihadapi dan jenis model yang digunakan. Fungsi aktivasi membantu memperkenalkan non-linearitas pada model sehingga dapat memodelkan hubungan yang lebih kompleks antara input dan output, yang pada akhirnya dapat meningkatkan kemampuan model dalam mempelajari pola-pola yang rumit dalam data.

XIII. Feed Forward Net

Feed Forward Network (FFN) adalah salah satu jenis arsitektur dasar dalam jaringan saraf tiruan (neural network). FFN terdiri dari serangkaian neuron yang terhubung secara sekuensial tanpa ada siklus atau umpan balik antara neuron-neuron tersebut.

Pada FFN, informasi mengalir melalui jaringan secara maju (forward) dari layer input ke layer output. Setiap neuron dalam layer tertentu menerima input dari neuron-neuron pada layer sebelumnya, melakukan operasi matematika pada input tersebut, dan menghasilkan output yang dikirim ke neuron-neuron pada layer berikutnya. Proses ini berlanjut hingga mencapai layer output, yang menghasilkan prediksi atau output akhir dari model.

Setiap neuron dalam FFN memiliki bobot dan fungsi aktivasi. Bobot digunakan untuk mengatur kontribusi input neuron terhadap outputnya, sedangkan fungsi aktivasi memperkenalkan non-linearitas ke dalam model. Biasaanya, FFN memiliki setidaknya satu hidden layer antara layer input dan output, di mana komputasi dan pemodelan non-linear terjadi.

FFN digunakan dalam berbagai tugas, seperti klasifikasi, regresi, pengenalan pola, dan pemrosesan bahasa alami. Dalam proses pelatihan, bobot dan bias dalam FFN diupdate melalui metode seperti backpropagation dan optimisasi gradient descent, untuk mencapai kemampuan model yang lebih baik dalam mempelajari pola-pola yang ada dalam data.

Dalam FFN, arsitektur dan jumlah neuron pada setiap layer, jenis fungsi aktivasi, serta parameter-parameter lainnya dapat disesuaikan dengan kebutuhan tugas yang dihadapi. Dengan menyusun layer-layer dan mengatur parameter-parameter tersebut, FFN dapat memodelkan hubungan yang kompleks dalam data dan melakukan prediksi yang akurat.

XIV. CNN (Convolutional Neural Network)

Convolutional Neural Network (CNN) adalah jenis arsitektur jaringan saraf tiruan yang secara khusus dirancang untuk memproses data berupa grid seperti gambar atau data spasial lainnya. CNN sangat efektif dalam mempelajari fitur-fitur visual kompleks dalam data gambar dan digunakan secara luas dalam berbagai tugas pengenalan gambar, penglihatan komputer, dan pengolahan citra.

CNN memiliki beberapa komponen utama, termasuk layer konvolusi, layer pooling, dan layer fully connected. Layer konvolusi adalah inti dari CNN, di mana filter atau kernel konvolusi diterapkan pada input untuk mendeteksi fitur-fitur visual seperti tepi, tekstur, dan pola lokal lainnya. Filter ini bergerak secara bertahap di seluruh input untuk menghasilkan peta fitur (feature map) yang menyoroti keberadaan fitur-fitur tersebut.

Layer pooling digunakan untuk mereduksi dimensi spasial dari peta fitur yang dihasilkan oleh layer konvolusi. Biasanya, operasi pooling dilakukan dengan mengambil nilai maksimum (max pooling) atau menghitung rata-rata (average pooling) dari sekelompok nilai dalam peta fitur. Tujuannya adalah untuk mengurangi ukuran representasi data dan menjaga invariansi terhadap pergeseran kecil dalam posisi fitur.

Setelah melalui serangkaian layer konvolusi dan pooling, fitur-fitur yang relevan diekstraksi dari gambar. Kemudian, output dari layer pooling diberikan ke layer fully connected, yang merupakan serangkaian neuron yang terhubung sepenuhnya dan bertanggung jawab untuk melakukan klasifikasi atau regresi pada fitur-fitur yang telah diekstraksi.

Selama proses pelatihan, bobot dan bias dalam CNN diperbarui melalui algoritma backpropagation dan optimisasi gradient descent, di mana gradien kesalahan dikalkulasikan dan digunakan untuk menyesuaikan parameter-parameter model.

Keunggulan CNN terletak pada kemampuannya dalam mengenali fitur-fitur lokal secara hierarkis dan melakukan ekstraksi fitur secara otomatis, tanpa memerlukan pengenalan fitur

manual. CNN telah mencatat kemajuan yang signifikan dalam pengenalan gambar, deteksi objek, segmentasi, dan tugas penglihatan komputer lainnya, dan digunakan secara luas dalam berbagai aplikasi seperti pengenalan wajah, pengenalan tulisan tangan, kendaraan otonom, dan banyak lagi.

XV. Transfer Learning

Transfer Learning adalah pendekatan dalam deep learning di mana model yang telah dilatih pada tugas tertentu digunakan sebagai awal untuk melatih model baru pada tugas yang berbeda. Ide dasar di balik transfer learning adalah bahwa pengetahuan yang telah diperoleh oleh model saat melatih pada tugas yang besar dan kompleks dapat digunakan kembali untuk meningkatkan kinerja model pada tugas yang berbeda, terutama ketika data pelatihan yang tersedia untuk tugas baru terbatas.

Dalam transfer learning, model yang telah dilatih sebelumnya, yang sering kali merupakan model yang besar dan kompleks seperti Convolutional Neural Network (CNN), digunakan sebagai "pengenal fitur" yang telah belajar mengekstraksi fitur-fitur penting dari data. Lapisan-lapisan awal model tersebut yang bertanggung jawab atas ekstraksi fitur umum biasanya dibekukan (frozen), yang berarti bobotnya tidak diubah selama pelatihan model baru. Kemudian, lapisan-lapisan akhir model yang terhubung ke tugas spesifik dilatih ulang (fine-tuned) menggunakan data pelatihan yang tersedia untuk tugas baru.

Transfer learning dapat memberikan beberapa manfaat. Pertama, itu dapat mengurangi kebutuhan akan jumlah data pelatihan yang besar untuk melatih model baru. Model yang telah dilatih sebelumnya telah mempelajari fitur-fitur umum yang berguna dan dapat digunakan pada tugas baru dengan data pelatihan yang lebih sedikit. Kedua, transfer learning dapat membantu menghindari overfitting pada model baru, karena model awal yang telah dilatih umumnya sudah memiliki kemampuan generalisasi yang baik.

Transfer learning telah menjadi pendekatan yang populer dalam deep learning dan telah berhasil digunakan dalam berbagai tugas seperti pengenalan gambar, deteksi objek, pemrosesan bahasa alami, dan lainnya. Dengan memanfaatkan pengetahuan yang telah diperoleh dari model yang sudah ada, transfer learning mempercepat dan meningkatkan kinerja pengembangan model dalam banyak kasus.

XVI. T**ensorboard**

TensorBoard adalah alat visualisasi yang disediakan oleh TensorFlow untuk membantu pemahaman dan pemantauan model dan pelatihan dalam deep learning. Ini menyediakan antarmuka grafis yang interaktif yang memungkinkan pengguna untuk melihat metrik pelatihan, visualisasi grafik komputasi, dan menganalisis performa model secara real-time.

Dengan TensorBoard, Anda dapat melacak dan memvisualisasikan metrik seperti loss, akurasi, dan kecepatan konvergensi selama pelatihan model. Ini memungkinkan Anda untuk memantau kemajuan pelatihan dan memeriksa apakah model Anda berkinerja dengan baik atau perlu disesuaikan.

Selain itu, TensorBoard juga menyediakan visualisasi grafik komputasi yang memungkinkan Anda untuk memeriksa arsitektur model secara visual. Anda

dapat melihat struktur jaringan, hubungan antara lapisan, dan aliran data melalui model Anda. Ini membantu dalam pemahaman yang lebih baik tentang model Anda dan memudahkan dalam pemecahan masalah dan debugging jika diperlukan.

Selain itu, TensorBoard juga dapat digunakan untuk memvisualisasikan representasi fitur dalam ruang rendah dimensi (embedding) menggunakan teknik seperti t-SNE (t-Distributed Stochastic Neighbor Embedding). Ini dapat membantu dalam pemahaman yang lebih baik tentang bagaimana model Anda mempelajari representasi fitur yang berguna.

TensorBoard merupakan alat yang sangat berguna dalam pengembangan dan analisis model deep learning. Dengan memanfaatkannya, Anda dapat memperoleh wawasan yang lebih baik tentang kinerja dan perilaku model Anda, yang dapat membantu Anda dalam pengambilan keputusan yang lebih baik dalam mengoptimalkan model Anda.

XVII. Save Load

Menyimpan dan memuat model adalah proses penting dalam pengembangan model machine learning. Saat model Anda telah dilatih, Anda ingin dapat menyimpannya untuk digunakan di masa depan atau untuk berbagi dengan orang lain. Dalam konteks deep learning, terdapat dua aspek utama yang terkait dengan menyimpan dan memuat model:

Menyimpan Model: Proses menyimpan model melibatkan menyimpan arsitektur model, parameter bobot, dan informasi lain yang diperlukan untuk merekonstruksi model secara lengkap. Dalam banyak kerangka kerja deep learning seperti TensorFlow dan PyTorch, ini biasanya dilakukan dengan menyimpan model ke dalam file yang dapat diakses nanti. Format yang umum digunakan termasuk TensorFlow SavedModel, Keras H5, PyTorch PT, dll. Dengan menyimpan model, Anda dapat menggunakan kembali model tersebut di masa depan tanpa perlu melatih ulang.

Memuat Model: Setelah model Anda disimpan, Anda dapat memuatnya kembali di waktu berikutnya untuk digunakan. Proses memuat model melibatkan membaca file yang berisi arsitektur dan parameter model, dan membangun kembali model tersebut dalam memori. Setelah model dimuat, Anda dapat menggunakannya untuk inferensi atau melanjutkan pelatihan jika perlu. Dalam kerangka kerja seperti TensorFlow dan PyTorch, biasanya ada fungsi atau metode yang disediakan untuk memuat model dengan mudah.

Proses menyimpan dan memuat model penting karena memungkinkan Anda untuk menghindari pelatihan ulang yang mahal secara komputasional dan waktu. Ini juga memungkinkan Anda untuk berbagi model dengan orang lain dan mempercepat proses pengembangan dan eksperimen dengan model deep learning.

Selain itu, ketika menyimpan dan memuat model, penting untuk memastikan kompatibilitas antara versi kerangka kerja yang digunakan saat menyimpan dan memuat model. Versi yang tidak kompatibel dapat menyebabkan masalah saat memuat model, jadi pastikan untuk

menggunakan versi yang sesuai saat menyimpan dan memuat model Anda.

XVIII. Kesimpulan

Dalam serangkaian pertanyaan di atas, kita telah membahas beberapa konsep dan teknik penting dalam machine learning dan deep learning. Berikut adalah kesimpulan dari setiap pertanyaan:

Dataset dan Dataloader: Dataset adalah kumpulan data yang digunakan untuk melatih dan menguji model. Dataloader adalah utilitas yang membantu memuat data dalam batch ke dalam model. Dataloader memungkinkan kita untuk membagi dataset menjadi batch-batch kecil dan mengatur shuffle data serta proses pemuatan data paralel.

Dataset Transforms: Dataset Transforms adalah teknik untuk mengubah data mentah dalam dataset menjadi bentuk yang lebih sesuai untuk pelatihan model. Ini dapat mencakup transformasi seperti normalisasi, pemangkasan, pergeseran, atau transformasi lain yang diterapkan pada data sebelum dimasukkan ke dalam model.

Autograd: Autograd adalah fitur di beberapa kerangka kerja deep learning yang memungkinkan perhitungan gradien otomatis. Ini memungkinkan kita untuk menghitung gradien dari fungsi yang didefinisikan dan menggunakannya untuk pelatihan model dengan algoritma seperti backpropagation.

Backpropagation: Backpropagation adalah algoritma yang digunakan untuk menghitung gradien dari fungsi kerugian terhadap parameter model. Dalam pelatihan model neural network, backpropagation menghitung gradien melalui serangkaian layer dan menggunakan gradien ini untuk memperbarui parameter agar model dapat belajar.

Gradient Descent: Gradient Descent adalah algoritma optimisasi yang digunakan dalam pelatihan model untuk mencari nilai parameter yang menghasilkan nilai fungsi kerugian yang minimal. Algoritma ini bergerak di arah gradien negatif fungsi kerugian dengan langkah-langkah kecil untuk mencapai minimum lokal atau global.

Softmax dan Cross Entropy: Softmax adalah fungsi aktivasi yang mengubah output dari model menjadi distribusi probabilitas yang menggambarkan kelas yang mungkin. Cross Entropy adalah fungsi kerugian yang digunakan dalam klasifikasi multikelas untuk membandingkan probabilitas yang diprediksi oleh model dengan probabilitas yang sebenarnya.

Activation Functions: Activation Functions adalah fungsi matematis yang diterapkan pada output neuron dalam jaringan saraf. Mereka memperkenalkan non-linearitas ke dalam model, yang memungkinkan model untuk mempelajari hubungan yang lebih kompleks dalam data. Contoh aktivasi yang umum digunakan termasuk ReLU, Sigmoid, dan Tanh.

Feed Forward Neural Network: Feed Forward Neural Network adalah arsitektur dasar dalam deep learning di mana informasi mengalir maju melalui layer-layer neuron. Setiap neuron menerima input, mengalikan bobot dengan input, menerapkan fungsi aktivasi, dan mengirim output ke layer berikutnya. Ini adalah model yang paling sederhana dalam deep learning.

Convolutional Neural Network (CNN): CNN adalah jenis arsitektur jaringan saraf yang khusus dirancang untuk tugas pengenalan gambar dan pengolahan citra. CNN menggunakan layer konvolusi untuk mengekstraksi fitur dari gambar secara hierarkis. Ini memungkinkan model untuk menangkap pola lokal dan mempertahankan invariansi spasial.

Transfer Learning: Transfer Learning adalah teknik di mana model yang telah dilatih pada tugas lain digunakan sebagai titik awal untuk melatih model baru pada tugas yang berbeda. Ini memanfaatkan pengetahuan yang telah dipelajari oleh model sebelumnya untuk mempercepat dan meningkatkan kinerja pelatihan pada tugas baru dengan dataset yang lebih kecil.

Tensorboard: Tensorboard adalah alat visualisasi yang disediakan oleh TensorFlow untuk memantau, menganalisis, dan memvisualisasikan pelatihan model. Ini memungkinkan kita untuk melihat grafik model, memantau metrik pelatihan seperti loss dan akurasi, serta memvisualisasikan histogram bobot dan gradien.

Save Load: Save Load adalah proses menyimpan dan memuat model yang telah dilatih. Dengan menyimpan model, kita dapat menggunakan kembali model tersebut di masa depan tanpa perlu melatih ulang. Ini memungkinkan kita untuk berbagi model dengan orang lain dan mempercepat proses pengembangan model.

Secara keseluruhan, topik-topik ini membentuk dasar penting dalam pemahaman dan pengembangan model machine learning dan deep learning. Memahami konsep-konsep ini akan membantu dalam merancang, melatih, dan mengoptimalkan model yang lebih baik.