JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Data Science: Artificial Intelligence (BU.920.624)

Group Assignment #2

Due on November 20th via Canvas

Group Members: Mohammed Bo Khamseen, Bonnie Koo, Pranab Sharma, Pooja Singh

1. Read the Harvard Business School case "The Future in Sight: LumineticsCore and the First Autonomous AI for Diagnostics"

(a) Explain how LumineticsCore works and how it was trained. Discuss why Dr. Abramoff's design and data strategy enabled him to secure FDA authorization ahead of competitors such as Google, drawing on the case's description of ground truth standards, dataset construction, and the autonomy requirement. Based on your own professional background and career interests, and if helpful the perspectives of your team members, critique the main strengths and limitations of his algorithmic and dataset choices.

Analysis

LumineticsCore works by using CNN to identify specific retinal features on ophthalmology images that are already well backed medically as indicators for DR diagnosis. These features were analyzed depending on the likelihood, size, shape, location etc. Then all these features were combined into the DR index, which had a threshold to determine whether this is a DR or not.

The reasons why he got ahead of other competitors

1. He found a metric that is medically accepted in the field (both clinical practice and clinical trials) as a consistent, objective ground truth. LumineticsCore used FPRC as the gold standard for grading diabetic retinopathy. This served as the ground truth for correct DR diagnosis - a much better benchmark than relying on doctors' diagnoses, which are known to disagree in up to 15% of cases.
2. He understood the importance of transparency and trust in medicine. Instead of a black box resulting from end-to-end CNN, he matched different AI models to the right functions so the process could be explained. CNNs did feature extraction from images. A separate fusion algorithm then reasoned out the DR index. This setup made it clear how the product worked and how the decisions were made. It solved the black box problem as CNNs just focused on feature identification, and how those features were used to come up with a diagnosis was transparent. Statistical analyses were also used to check for bias.
3. More importantly, decision-making by the system was guided by established medical theory, rather than letting AI decide or diagnose based on some new unknown factors. Although LumineticsCore operated autonomously, its design was firmly grounded in peer-reviewed medical theory. This made it easier for physicians to understand and trust autonomous diagnoses, as the product reduced ambiguity and limits the need for subjective judgment, helping bridge the gap between AI and clinical acceptance.
4. For the autonomy requirement, the model aimed for higher sensitivity and specificity compared to human judgment, directly tackling the risk of inaccuracy and potential distrust of the product. Also, the product was specifically indicated for more than mild DR, meaning that it was built to catch moderate and severe DR cases. This meant that the implication of overdiagnosis was less severe compared to underdiagnosis in mild cases.

Strengths:

- Explainable model
- Used an objective clinical metric that is widely accepted by the medical community as the ground truth
- Used existing medical theory to form decision making processes

Limitations

- Only indicated for more than mild DR. Although moderate and severe DR detection is prioritized, missing early stages may result in delayed intervention in the bigger scheme of DR care pathway.
- Even though high sensitivity and specific rates are high, there is no clarity around how to deal with the consequences of false positives.
- Unsure whether the image quality could have an impact on the diagnosis quality
- Possibility that FPRC grading isn't the ground truth
- Unsure about the type of data that was collected to form the model and whether other bias other than racial, ethnic, sex bias was clearly addressed.

(b) Describe the pivotal trial and evaluation method used for FDA clearance, including the target condition, site selection, patient sample, performance thresholds, diagnosability rate, and the use of sensitivity and specificity as primary endpoints. Drawing from your own background and those of your teammates, explain why these metrics were chosen and how this evaluation approach aligns or conflicts with the priorities of patients, primary care physicians, ophthalmologists, the FDA, and payers, using concrete examples from the case.

Analysis

1. **900 people with diabetes with no prior diagnosis of diabetic retinopathy:**
    - This is the right target population (people at risk, but no prior DR) making it a solid test of the model's ability to generalize to "unknown" patients. However, some clinicians raised concerns about the consistency of performance across a diverse patient population and may question whether 900 patients are sufficient to address that diversity.
2. **Primary care sites in the US:**
    - The trial recruited from primary care, which reflects the intention of this product trying to address that only about 15% of US diabetes patients get annual specialist screening. This approach is helpful in targeting the broader diabetes population since many patients are generally managed by PCPs in community settings. However, patients seen in primary care may differ from those treated in specialty clinics, potentially affecting the model's generalizability and making ophthalmologists skeptical about applying the tool in their own practices. As a result, there were suggestions to limit its use to a triage function (not as the first-line diagnostic tool) and/or deployment within the PCP environment.
3. **Performance results were 87.2% sensitivity, 90.7% specificity exceeding the predefined thresholds of 85% and 82.5%**:
    - Using sensitivity and specificity as primary endpoints is appropriate since the product's core function is accurate diagnosis. The FDA has intentionally set high predefined thresholds to ensure a margin of error and instill trust in the product, aligning with its priorities of safety and effectiveness. However, from a PCP perspective, there are practical concerns about liability or malpractice if the product makes an incorrect diagnosis. At the same time, this risk of potential misdiagnosis supports the role of ophthalmologists in managing complex cases where clinical judgment is still required. This perspective aligns with the ophthalmologists' suggestion to use the tool as a second opinion or as a triage mechanism to be deployed within PCPs
4. **Diagnosability rate of 96.1%:**
    - This means that most operators can perform the diagnosis without prior knowledge or specialized skills. However, since the primary endpoints focused on diagnostic accuracy rather than metrics such as time to diagnosis or overall care pathway improvements due to high diagnosability rate, payers may have found it challenging to assign high value to the product. This, in turn, affected physician incentives. CMS reimbursement for physician-

provided diabetic retinopathy screening is at least twice that for screenings performed by the AI product, meaning there are no financial incentives for ophthalmologists to adopt the technology. Furthermore, even if physicians were open to adoption, new imaging equipment is required, which introduces change management and financial challenges, complicating integration into care practices. Without clear ROI of financial and effort investment, adoption could be challenging,

(c) If you are tasked with rebuilding LumineticsCore today, outline how you would redesign its clinical role (autonomous diagnosis versus triage versus decision support), technical architecture, regulatory strategy (de novo versus 510(k), trial design), and business/reimbursement model. Feel free to incorporate your own and your team members' professional experience. Show how your redesign would address at least two adoption obstacles highlighted in the case.

Analysis - Components for rebuilding Luminetics today

Clinical Role: I would shift LumineticsCore from a pure autonomous diagnostic tool to an integrated triage and decision-support system.

- Redesigned Role: Autonomous triage (Tier 1) + physician-in-the-loop decision support (Tier 2/3).
- Why: Reduces liability fears, aligns with clinical workflows, and strengthens trust by positioning AI as an extender-not a replacement.
- Adoption Obstacle Solved: Clinician skepticism & trust issues.

Technical Architecture - The system would be rebuilt as a modular, interpretable pipeline rather than a black-box model. Separate components would handle image quality, lesion detection, and risk scoring (including DRSS) with heatmaps and confidence levels shown directly to clinicians.

It will also support continuous model-accuracy monitoring, allowing transparent updates over time for post-market surveillance (aligned with FDA PCCP). Clinics will be able to see how accuracy changes, detect drift early, and adjust workflows accordingly.

- Why: Improves transparency, supports continuous updates (via FDA PCCP), and simplifies clinic operations with better image capture and workflow fit.
- Adoption Obstacle Solved: Skepticism about black-box AI and concerns over model drift.

Regulatory Strategy - Rather than repeating the original high-stakes de novo approval for "*autonomous diagnosis,"* I would pursue a de novo submission that emphasizes triage and decision support, supported by a PCP to allow pre-approved updates

- Trial Design: Non-inferiority for referral decisions + superiority in workflow efficiency and screening adherence; multi-site primary care trials to support coverage decisions. Conducting economic evaluation studies will also support regulatory approvals – such as measuring (Time to treatment (vs. Current status quo), incidence of diagnosis (vs. Status quo)
- Why: Faster iterations, regulatory flexibility, and evidence targeting workflow and payer value—not just accuracy.

Business & Reimbursement Model:

Commented [BK1]: potential other business models:

Retail Pharmacies: Could offer diabetic retinopathy screening using CPT codes, generating additional revenue by expanding access in community settings, provided reimbursement policies allow.

Treatment Identifier for Ophthalmologists: Position LumineticsCore as a tool to identify patients needing treatment, enhancing specialist workflows by prioritizing referrals and supporting clinical decisions.

Research Biomarker for Pharma Trials: Use LumineticsCore as an alternative to Fundus Photography Reading Centers for standardized, cost-effective diabetic retinopathy grading in clinical trials, aiding pharma drug development.

(1) Value-Based and Subscription-Driven - The business model would move away from dependence on a single CPT code, which was a major barrier in the original product

- Redesigned Model: Hybrid FFS + value-based reimbursement using existing CPT codes; payer-aligned per-patient-per-month incentives for improved DR screening adherence; subscription/SaaS pricing instead of high upfront costs.
- Why: Creates immediate billability, aligns with payer goals, and lowers adoption barriers for clinics.
- Adoption Obstacle Solved: Reimbursement and business model uncertainty.

(2) Retail Pharmacies: Could offer diabetic retinopathy screening using CPT codes, generating additional revenue by expanding access in community settings, provided reimbursement policies allow.

(3) Treatment Identifier for Ophthalmologists: Position LumineticsCore as a tool to identify patients needing treatment, enhancing specialist workflows by prioritizing referrals and supporting clinical decisions.

(4) Research Biomarker for Pharma Trials: Use LumineticsCore as an alternative to Fundus Photography Reading Centers for standardized, cost-effective diabetic retinopathy grading in clinical trials, aiding pharma drug development.

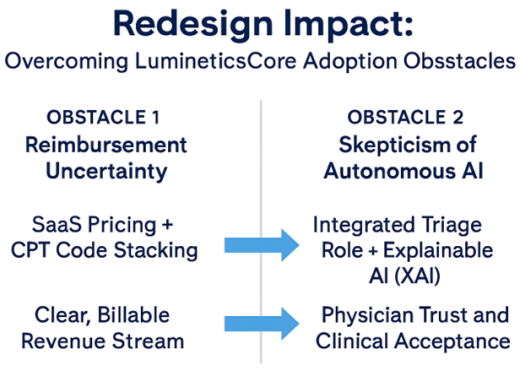
Tacking adoption challenges: These changes directly tackle two core adoption failures from the case: the lack of a viable reimbursement pathway, and deep skepticism toward fully autonomous AI.

Reimbursement & Business Model Uncertainty

- Hybrid FFS/value-based reimbursement + PPPM incentives create predictable revenue.
- SaaS pricing removes capital expense and simplifies adoption.

Skepticism & Trust in Autonomous AI

- Integrated triage role preserves physician oversight where needed.
- Explainable AI outputs build transparency and confidence.



(d) Propose at least three questions you would ask Dr. Abramoff. Aim for questions that reflect the background and interests of your team. For each question, briefly explain why it matters.

Analysis - Proposed Questions for Dr. Abramoff

Question 1 - *"If you were rebuilding LumineticsCore today, how would you redesign both the technical architecture and the product's positioning to improve explainability, workflow fit, and long-term performance monitoring in real clinical environments?"* (AI Engineering background - Pranab Sharma)

- Why: This question addresses the core technical challenge of modern AI in medicine: balancing the need for continuous improvement (learning from real-world data) with the regulatory requirement for a fixed, validated product. The answer would reveal Dr. Abramoff's strategy for MLOps and regulatory compliance in an adaptive system.

Question 2 - *"How do you deal with medical liability issues relating to false negatives of the autonomous diagnosis? How are the results presented to the clinicians?"* Care Delivery and Wider care system integration (Pharmacy and Clinical Background – Bonnie Koo)

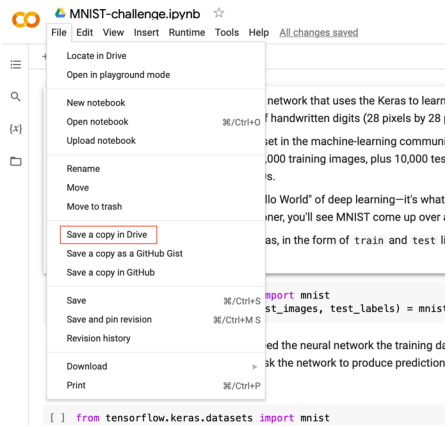
- Why: When doctors make incorrect judgments or misdiagnoses, medical liability generally falls primarily to the clinician and the healthcare provider. Physicians are professionally responsible for exercising sound clinical judgment based on the information available, including diagnostic tools, clinical guidelines, and patient history.

Question 3 - *"Which early decisions around reimbursement, workflow design, or stakeholder alignment most limited LumineticsCore's adoption, and what would you change now to accelerate scale across primary care networks?"* Workflow redesign, alignment with stakeholder (Tech strategy and Operations Consulting background - Mohammed Bo Khamseen + Pooja Singh)

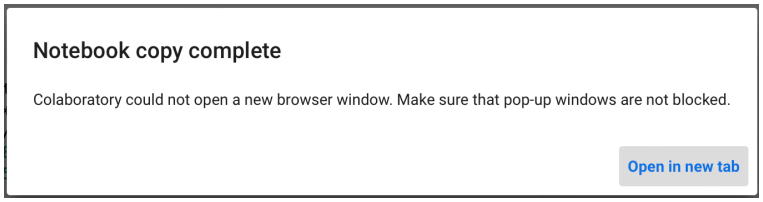
- Why it matters: his question targets the **business and operational challenges** that hindered adoption. A Digital Health Analyst understands that clinical efficacy is not enough; the system must demonstrate a clear Return on Investment (ROI) and seamless integration into the existing clinical workflow to secure large-scale adoption and favorable reimbursement. team is focused on digital health adoption, business models, and implementation. This question uncovers practical lessons about scaling AI in complex healthcare systems.

2. Open the Google Colab notebook "MNIST-challenge.ipynb" by clicking on the following link: https://colab.research.google.com/drive/1An8WZ2pj7NOFChDouMkOWuk0GtjF0Dvp?usp=sharing

After you've opened the above notebook, go to the "File" menu, and select "Save a copy in Drive" (as shown in the screenshot below):



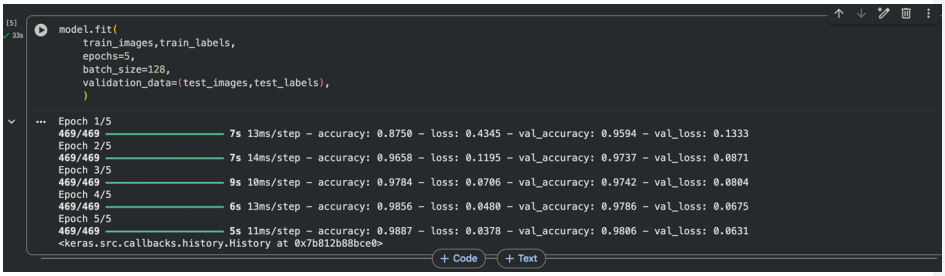
When you've done saving a copy to Drive, select "Open in new tab":



The notebook is now yours!

(a) Read and run the code in its entirety. How accurate is the model on the training data? How good is the model on the test data? Is the model prone to overfitting?

Analysis



As we reviewed the training output from the MNIST model, we observed that it performed very well on both the training and test sets. After five epochs, the model reached a training accuracy of 98.87% with a training loss of 0.0378. Its performance on unseen data was similarly strong, with a test (validation) accuracy of 98.06% and a validation loss of 0.0631. When we compare these numbers, the gap between the training and validation losses remains small across epochs, which indicates that the model is generalizing effectively and not simply memorizing the data. Because the validation loss steadily decreases and stays close to the training loss, we do not see evidence of meaningful overfitting. Overall, the model demonstrates strong learning performance and maintains high accuracy on new data.

(b) In the last section of your Google Colab notebook:

```
model.fit(
    train_images,train_labels,
    epochs=5,
    batch_size=128,
    validation_data=(test_images,test_labels),
    )
```

change "`epochs=5,`" to "`epochs=20,`".

Run the code again. How accurate is the newly trained model on the training data? How accurate is the newly trained model on the test data? Does this model still have an overfitting problem? If so, is the situation better or worse?

```
Epoch 1/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - accuracy: 0.9920 - loss: 0.0282 - val_accuracy: 0.9812 - val_loss: 0.0640
Epoch 2/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 6s 13ms/step - accuracy: 0.9944 - loss: 0.0201 - val_accuracy: 0.9796 - val_loss: 0.0696
Epoch 3/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - accuracy: 0.9958 - loss: 0.0160 - val_accuracy: 0.9813 - val_loss: 0.0621
Epoch 4/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 6s 13ms/step - accuracy: 0.9971 - loss: 0.0115 - val_accuracy: 0.9822 - val_loss: 0.0600
Epoch 5/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 0.9979 - loss: 0.0089 - val_accuracy: 0.9826 - val_loss: 0.0633
Epoch 6/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - accuracy: 0.9990 - loss: 0.0055 - val_accuracy: 0.9813 - val_loss: 0.0685
Epoch 7/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 6s 13ms/step - accuracy: 0.9993 - loss: 0.0043 - val_accuracy: 0.9823 - val_loss: 0.0603
Epoch 8/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - accuracy: 0.9994 - loss: 0.0033 - val_accuracy: 0.9837 - val_loss: 0.0636
Epoch 9/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 6s 13ms/step - accuracy: 0.9998 - loss: 0.0021 - val_accuracy: 0.9839 - val_loss: 0.0613
Epoch 10/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 0.9998 - loss: 0.0014 - val_accuracy: 0.9833 - val_loss: 0.0631
Epoch 11/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - accuracy: 0.9999 - loss: 0.0010 - val_accuracy: 0.9833 - val_loss: 0.0678
Epoch 12/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 10ms/step - accuracy: 1.0000 - loss: 7.2115e-04 - val_accuracy: 0.9839 - val_loss: 0.0630
Epoch 13/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - accuracy: 1.0000 - loss: 4.5990e-04 - val_accuracy: 0.9839 - val_loss: 0.0644
Epoch 14/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - accuracy: 1.0000 - loss: 4.1624e-04 - val_accuracy: 0.9841 - val_loss: 0.0641
Epoch 15/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 6s 13ms/step - accuracy: 1.0000 - loss: 3.3527e-04 - val_accuracy: 0.9839 - val_loss: 0.0656
Epoch 16/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 1.0000 - loss: 2.9167e-04 - val_accuracy: 0.9848 - val_loss: 0.0652
Epoch 17/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 5s 10ms/step - accuracy: 1.0000 - loss: 2.6445e-04 - val_accuracy: 0.9844 - val_loss: 0.0660
Epoch 18/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 6s 13ms/step - accuracy: 1.0000 - loss: 2.3786e-04 - val_accuracy: 0.9844 - val_loss: 0.0669
Epoch 19/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - accuracy: 1.0000 - loss: 2.0518e-04 - val_accuracy: 0.9842 - val_loss: 0.0673
Epoch 20/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - accuracy: 1.0000 - loss: 1.9953e-04 - val_accuracy: 0.9842 - val_loss: 0.0669
<keras.src.callbacks.history.History at 0x7b81baadb2c0>
```

After increasing the training schedule to 20 epochs, we noticed that the model becomes even more accurate

on the training data, eventually reaching 100% training accuracy with an extremely small loss (~0.0002). However, the test accuracy plateaus at around 98.4%, and the validation loss remains close to 0.06, which is significantly higher than the training loss. Compared to the 5-epoch model, the gap between training and test performance has widened, indicating stronger overfitting. In other words, while the model has effectively memorized the training data, its ability to generalize to unseen data has not improved and the overfitting problem is slightly worse than before.

(c) Propose and implement strategies to deal with overfitting. You can use the dropout technique, for example, by changing the Python code below "Let's build the network":

```
model=keras.Sequential([
 layers.Dense(512,  activation = "relu"),
 layers.Dense(10, activation = "softmax")
])
```

to

```
model=keras.Sequential([
  layers.Dense(512,  activation = "relu"),
  layers.Dropout(0.5),
  layers.Dense(10, activation = "softmax")
])
```

Feel free to change the dropout rate to any value you like. In addition to the dropout technique, consider increasing the size of your neural network (e.g., by changing the number of neurons from "512" to "784") or the number of hidden layers (e.g., by adding another layer using `layers.Dense(16, activation="relu"))`. You can also use the regularization technique (L1/L2).

Run your revised code and demonstrate that you have a better model in the sense that overfitting is less of a problem.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
model=keras.Sequential([
layers.Dense(768, activation = "relu"),
layers.Dropout(0.5),
layers.Dense(10, activation = "softmax")
])
```

```
model.fit(
    train_images,train_labels,
    epochs=20,
    batch_size=128,
    validation_data=(test_images,test_labels),
    )
```

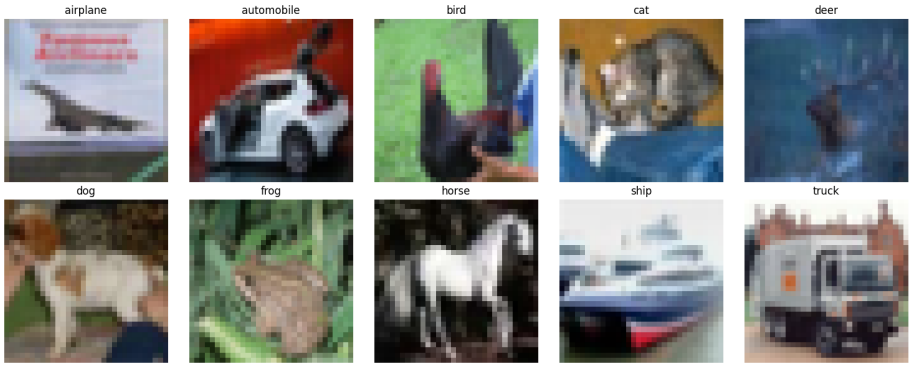
```
Epoch 1/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 20ms/step - accuracy: 0.8545 - loss: 0.4788 - val_accuracy: 0.9585 - val_loss: 0.1429
Epoch 2/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 18ms/step - accuracy: 0.9561 - loss: 0.1453 - val_accuracy: 0.9716 - val_loss: 0.0985
Epoch 3/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 11s 20ms/step - accuracy: 0.9682 - loss: 0.1067 - val_accuracy: 0.9742 - val_loss: 0.0848
Epoch 4/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 20ms/step - accuracy: 0.9748 - loss: 0.0835 - val_accuracy: 0.9777 - val_loss: 0.0737
Epoch 5/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 8s 17ms/step - accuracy: 0.9761 - loss: 0.0755 - val_accuracy: 0.9796 - val_loss: 0.0638
Epoch 6/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 18ms/step - accuracy: 0.9791 - loss: 0.0711 - val_accuracy: 0.9808 - val_loss: 0.0626
Epoch 7/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 21ms/step - accuracy: 0.9825 - loss: 0.0555 - val_accuracy: 0.9819 - val_loss: 0.0607
Epoch 8/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 21ms/step - accuracy: 0.9832 - loss: 0.0524 - val_accuracy: 0.9820 - val_loss: 0.0612
Epoch 9/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 18ms/step - accuracy: 0.9845 - loss: 0.0475 - val_accuracy: 0.9833 - val_loss: 0.0589
Epoch 10/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 21ms/step - accuracy: 0.9865 - loss: 0.0428 - val_accuracy: 0.9819 - val_loss: 0.0588
Epoch 11/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 20ms/step - accuracy: 0.9872 - loss: 0.0400 - val_accuracy: 0.9836 - val_loss: 0.0579
Epoch 12/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 19ms/step - accuracy: 0.9872 - loss: 0.0380 - val_accuracy: 0.9836 - val_loss: 0.0569
Epoch 13/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 20ms/step - accuracy: 0.9890 - loss: 0.0335 - val_accuracy: 0.9826 - val_loss: 0.0619
Epoch 14/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 20ms/step - accuracy: 0.9901 - loss: 0.0307 - val_accuracy: 0.9843 - val_loss: 0.0605
Epoch 15/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 20ms/step - accuracy: 0.9906 - loss: 0.0309 - val_accuracy: 0.9833 - val_loss: 0.0586
Epoch 16/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 18ms/step - accuracy: 0.9911 - loss: 0.0274 - val_accuracy: 0.9841 - val_loss: 0.0587
Epoch 17/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 21ms/step - accuracy: 0.9915 - loss: 0.0251 - val_accuracy: 0.9836 - val_loss: 0.0608
Epoch 18/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 20ms/step - accuracy: 0.9923 - loss: 0.0230 - val_accuracy: 0.9833 - val_loss: 0.0591
Epoch 19/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 20ms/step - accuracy: 0.9926 - loss: 0.0237 - val_accuracy: 0.9847 - val_loss: 0.0589
Epoch 20/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 17ms/step - accuracy: 0.9928 - loss: 0.0232 - val_accuracy: 0.9846 - val_loss: 0.0587
<keras.src.callbacks.history.History at 0x78f23a1a5eb0>
```

We modified the network architecture to address overfitting as suggested, and we observed a clear improvement in how well the model generalised to unseen data. We increased the size of the hidden layer and added dropout. After retraining the model for 20 epochs, training accuracy reached 99.3% and test accuracy reached 98.5%. More important to note is that both losses moved together across the epochs, which shows improvement over previous model where only training loss reached almost 0 and created a large gap with validation loss, demonstrating overfitting. Now the validation loss decreased below 0.06, showing a smaller gap with training loss.

3. Use a generative AI tool, such as ChatGPT, Claude.ai, or Google Gemini, to help you build and train an AI model in Google Colab. You may choose one MedMNIST dataset (PneumoniaMNIST, BreastMNIST, DermaMNIST, or PathMNIST), or, if you prefer a non-health application, use CIFAR-10 or CIFAR-100. With guidance from your GenAI assistant, write Colab-ready code that automatically loads your chosen dataset, builds an initial model, and then iteratively improves it to increase accuracy. In your submission, explain what your model aims to accomplish and why it is useful, provide your full implementation, describe how you used the GenAI tool during development, report your final test accuracy, and offer a concise reflection on which model changes led to the greatest improvement

Analysis:

Model Aim and Usefulness: The project's goal is to build a CNN to identify objects in small color images from the CIFAR-10 dataset. The model's aim is to look at a new image and correctly predict which of the 10 categories it belongs to. The 10 classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.



This is useful for a basic system for identifying passing vehicles (automobiles, trucks) or animals (birds, cats, deer, dogs, frogs, horses) on a road.

Full Implementation: Below is the link to the Colab code I used to load the data, build the initial model, and then build the improved model.

Google Colab File

I used Google Gemini as my gen AI assistant throughout this project as it was already in the Google Colab page. My process was:

Here is the Model Structure I used at the end.

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 30, 30, 32)	896
batch_normalization (BatchNormalization)	(None, 30, 30, 32)	128
max_pooling2d_4 (MaxPooling2D)	(None, 15, 15, 32)	0
dropout (Dropout)	(None, 15, 15, 32)	0
conv2d_5 (Conv2D)	(None, 13, 13, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 13, 13, 64)	256
max_pooling2d_5 (MaxPooling2D)	(None, 6, 6, 64)	0
dropout_1 (Dropout)	(None, 6, 6, 64)	0
flatten_2 (Flatten)	(None, 2304)	0
dense_4 (Dense)	(None, 128)	295,040
dropout_2 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 10)	1,290

Total params: 316,106 (1.21 MB)
Trainable params: 315,914 (1.21 MB)
Non-trainable params: 192 (768.00 B)

Use of Generative AI (Google Gemini)

I used Google Gemini as my gen AI assistant throughout this project as it was already in the Google Colab page. My process was:

1. **Initial Setup:** I first asked Gemini to help me write the code to load the CIFAR-10 dataset and build an initial CNN model.
2. **Preprocessing:** Gemini built the model and went to preprocessing the data, specifically normalizing the pixel values (dividing by 255.0) and one-hot encoding the labels so the model could understand them.
3. **Analysis & Iteration:** After the first model was trained, I shared the results with Gemini. Training accuracy was high (85%), but the validation accuracy was stuck at 69.8%. This was a clear case of overfitting, the model was memorizing the training images but wasn't learning to generalize to new ones.
4. **Proposing Improvements:** After asking Gemini to improve the model based on the results, Gemini suggested that to fix overfitting, I should use "regularization" techniques. It specifically recommended adding Batch Normalization layers to stabilize learning and Dropout layers to prevent the model from relying too much on any single neuron.
5. **Refinement:** It then helped me write the code for the new, improved model that included these Dropout and Batch Normalization layers.

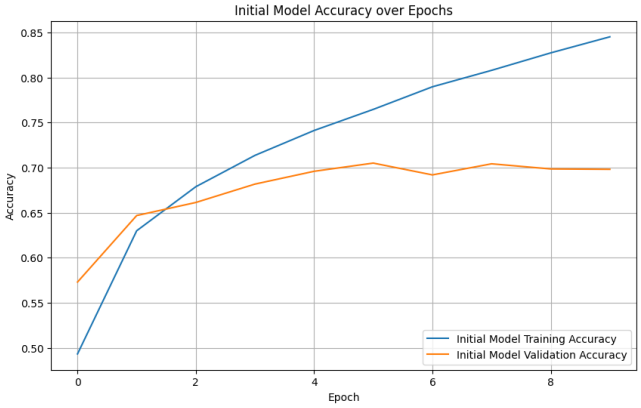
Final Test Accuracy: After training and testing both models on the test dataset, I got the following results:

- **Initial Model Test Accuracy:** 70%%
- **Final Improved Model Test Accuracy:** 71%

The iterative improvements successfully increased the model's accuracy on new data. But the real benefit was how much the training accuracy and testing accuracy were more alike. See below for graphs

Reflection on Model Improvements

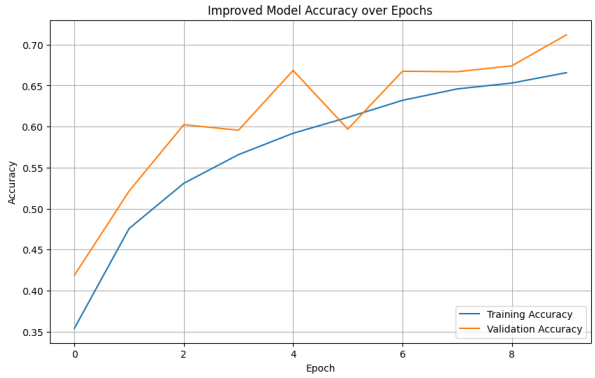
The single greatest improvement came from addressing overfitting.
My first model learned the training data too well. The validation accuracy was much lower than the training accuracy, and the validation loss started to increase, which meant the model was just memorizing. As the graph for the initial model shows, the training accuracy kept climbing while the validation accuracy plateaued.



The changes that led to the best improvement were adding Dropout and Batch Normalization layers.

- **Dropout (0.25 and 0.5):** Randomly turning off a fraction of neurons during each training step. This forced the network to learn more general features instead of just memorizing details.
- **Batch Normalization:** This helped stabilize the training process by normalizing the outputs of each layer.

The graph for the improved model shows a better trend:



By Adding these layers, the improved model's training accuracy (67%) was lower than the first model's, but its validation accuracy (71%) was higher. As seen in the second graph, the two lines are much closer together. This shows the gap between training and testing is smaller, meaning less overfitting.

4. Consider the following two-dimensional 5 × 5 input feature map:

0	1	1	1	0
0	-1	0	-1	1
0	0	-1	0	1
0	-1	0	-1	1
0	0	0	0	0

We now apply Convolution, Activation (ReLU) and MaxPooling to it.

(a) First, we apply the following two-dimensional, 2 × 2 convolutional filter

1	0
0	1

to the input feature map. Represent the output feature map.

Answer

Assuming stride=1 as its the convention
Output shape is (5-2+1)*(5-2+1)=4*4 matrix

For row 1-

1*0 + 0*1 + 0*0 + 1*(-1) = 0 + 0 + 0 + (-1) = -1
1*1 + 0*1 + 0*(-1) + 1*0 = 1 + 0 + 0 + 0 = 1
1*1 + 0*1 + 0*0 + 1*(-1) = 1 + 0 + 0 + (-1) = 0
1*1 + 0*0 + 0*(-1) + 1*1 = 1 + 0 + 0 + 1 = 2

For row 2-

1*0 + 0*(-1) + 0*0 + 1*0 = 0 + 0 + 0 + 0 = 0
1*(-1) + 0*0 + 0*0 + 1*(-1) = -1 + 0 + 0 + (-1) = -2
1*0 + 0*(-1) + 0*(-1) + 1*0 = 0 + 0 + 0 + 0 = 0
1*(-1) + 0*1 + 0*0 + 1*1 = -1 + 0 + 0 + 1 = 0

For row 3-

1*0 + 0*0 + 0*0 + 1*(-1) = 0 + 0 + 0 + (-1) = -1
1*0 + 0*(-1) + 0*(-1) + 1*0 = 0 + 0 + 0 + 0 = 0
1*(-1) + 0*0 + 0*0 + 1*(-1) = -1 + 0 + 0 + (-1) = -2
1*0 + 0*1 + 0*(-1) + 1*1 = 0 + 0 + 0 + 1 = 1

For row 4-

1*0 + 0*(-1) + 0*0 + 1*0 = 0 + 0 + 0 + 0 = 0
1*(-1) + 0*0 + 0*0 + 1*0 = -1 + 0 + 0 + 0 = -1
1*0 + 0*(-1) + 0*0 + 1*0 = 0 + 0 + 0 + 0 = 0
1*(-1) + 0*1 + 0*0 + 1*0 = -1 + 0 + 0 + 0 = -1

Output:

[-1 1 0 2]
[0 –2 0 0]
[-1 0 –2 1]
[0 –1 0 –1]

(b) Apply the ReLU activation function to the result from step (a). Represent the output feature map.

[0 1 0 2]
[0 0 0 0]
[0 0 0 1]
[0 0 0 0]

(c) Maxpool the result from step (b) with 2 × 2 filters and stride 2. Represent the output feature map.

Stride=2 so for a 4*4 matrix the maxpooled output will be 2*2

[1 2]

[0 1]