TRẢ LỜI CÂU HỎI SEMINAR

Đề tài: ML-1 (Keras)
Nhóm thực hiện: GROUP CQ2021-01

Tổng số câu hỏi: 25
Phân loại thành: 10 nhóm

DANH SÁCH CÂU HỎI

A. **Nhóm câu hỏi liên quan đến Model deployment**

1. Once trained, how can you effectively serialise a Keras model for deployment in production environments?
2. Làm thế nào để lưu trữ và tải lại một mô hình Keras đã huấn luyện?

B. **Nhóm câu hỏi liên quan đến mô hình được hỗ trợ bởi Keras**

3. Keras hỗ trợ những loại mô hình học sâu nào?
4. Một vài mô hình có thể sử dụng trong Keras?

C. **Nhóm câu hỏi liên quan đến dữ liệu lớn**

5. Keras có những phương pháp nào để xử lý dữ liệu lớn hiệu quả trong quá trình huấn luyện mô hình học sâu?
6. Keras có hỗ trợ các hệ thống dữ liệu lớn không?
7. Keras có hoạt động tốt trên bộ dữ liệu lớn không?

D. **Nhóm câu hỏi liên quan đến Debugging trong Keras**

8. Có cách nào để debug khi gặp low-level errors trong Keras không?
9. Mình muốn biết về cách debug trong Keras
10. Khả năng debug model dùng Sequential trong Keras như thế nào?

E. **Nhóm câu hỏi liên quan đến sự khác nhau giữa các mô hình: Sequential Model vs Functional Model**

11. Mình có một câu hỏi là sự khác biệt giữa Sequential API và Functional API trong Keras là gì? Nhóm có thể giải thích chi tiết hơn không?
12. Sequential API vs Functional API ?

F. **Nhóm câu hỏi liên quan đến huấn luyện phân tán dữ liệu**

13. Can we train LLM such as Llama Meta on a distributed dataset?
14. Keras có hỗ trợ cho việc huấn luyện phân tán (distributed training) không? Nếu có thì làm như thế nào?

G. **Nhóm câu hỏi liên quan đến so sánh Keras với các framework khác**

15. Trong các usecase nào việc sử dụng Keras thì vượt trội hơn so với các thư viện deep learning khác? Tại sao?
16. Tốc độ training khi train các mô hình deep learning giữa các thư viện với keras?
17. Tại sao tensorflow lại không thân thiện với beginner như Keras?

H. **Nhóm câu hỏi liên quan đến tiền xử lý dữ liệu với Keras**

* Chú ý: Ngôn ngữ trả lời là tiếng Anh hay tiếng Việt tùy thuộc vào ngôn ngữ được sử dụng trong câu hỏi.

18. Keras có khả năng tạo data augmentation tốt không?
19. Việc xử lý các điểm dữ liệu ở góc, cạnh của ảnh sẽ được xử lý như thế nào?
20. Làm thế nào để xử lý dữ liệu thời gian (time series) bằng Keras?
21. Kiểu nếu chúng ta chỉ sử dụng ML thì để nâng cao hiệu suất sẽ cần thêm những tiền xử lý hoặc thêm những biến hỗ trợ. Liệu deep learning có cần nâng cao hiệu suất bằng cách tương tự hay không?

I. **Nhóm câu hỏi liên quan đến Demo**

22. Sự khác biệt cụ thể giữa tập test và tập validation trong demo?

K. **Các câu hỏi khác**

23. Keras có hỗ trợ visualize model hay không?
24. Can Keras integrate with other programming languages?
25. Công việc của lớp "Flatten" trong Keras là gì?

A. Nhóm câu hỏi liên quan đến Model deployment

1. **Once trained, how can you effectively serialise a Keras model for deployment in production environments?**

Keras supports save, serialise of models, although deployment will depend on other factors such as the platform that you want to deploy your model on.

For serialisation, you can save and load the model very easily using save(path), and load_model(path) to save and load model respectively. Here the example code:

Saving model:

```
model = ...  # Get model (Sequential, Functional Model, or Model
subclass)
model.save('path/to/location.keras')  # The file needs to end with the
.keras extension
```

Loading model:

```
model = keras.models.load_model('path/to/location.keras')
```

If you want to do more complex operations regarding serialising models such as saving weights, saving in JSON/yaml format... You can read more about that here.

For deployment, as I mentioned, it's going to depend on the platform. The most popular choice for model deployment for Keras models are TF-Serving, TF.js and TFLight - both because of Keras deep integration with TensorFlow and TensorFlow mature deployment & production tools.

2. **Làm thế nào để lưu trữ và tải lại một mô hình Keras đã huấn luyện?**

Như đã đề cập ở câu hỏi trên, chúng ta có thể lưu trữ và tải lại mô hình Keras đã huấn luyện như sau:

Saving model:

```
model = ...  # Get model (Sequential, Functional Model, or Model
subclass)
model.save('path/to/location.keras')  # The file needs to end with the
.keras extension
```

Loading model:

```
model = keras.models.load_model('path/to/location.keras')
```

B. Nhóm câu hỏi liên quan đến mô hình được hỗ trợ bởi Keras

3. **Keras hỗ trợ những loại mô hình học sâu nào?**
4. **Một vài mô hình có thể sử dụng trong Keras?**

Keras hỗ trợ người dùng xây dựng một số mô hình học máy đơn giản như CNN, RNN, transformer... và có cung cấp một số các mô hình pre-train như Xception, MobileNet, ResNet, Efficient Net... Xem danh sách chi tiết các model có sẵn tại [tại đây](#).

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4
MobileNetV2	14	71.3%	90.1%	3.5M	105	25.9	3.8
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet169	57	76.2%	93.2%	14.3M	338	96.4	6.3
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7
NASNetMobile	23	74.4%	91.9%	5.3M	389	27.0	6.7
NASNetLarge	343	82.5%	96.0%	88.9M	533	344.5	20.0
EfficientNetB0	29	77.1%	93.3%	5.3M	132	46.0	4.9
EfficientNetB1	31	79.1%	94.4%	7.9M	186	60.2	5.6
EfficientNetB2	36	80.1%	94.9%	9.2M	186	80.8	6.5
EfficientNetB3	48	81.6%	95.7%	12.3M	210	140.0	8.8
EfficientNetB4	75	82.9%	96.4%	19.5M	258	308.3	15.1

C. Nhóm câu hỏi liên quan đến dữ liệu lớn

5. **Keras có những phương pháp nào để xử lý dữ liệu lớn hiệu quả trong quá trình huấn luyện mô hình học sâu?**

Keras cung cấp một số phương pháp để xử lý dữ liệu lớn hiệu quả trong quá trình huấn luyện mô hình học sâu. Một số phương pháp này bao gồm:

- **Data generators**: Keras cho phép sử dụng data generators, đây là các hàm sinh ra các batch dữ liệu ngay lập tức trong quá trình huấn luyện. Điều này rất hữu ích khi bộ dữ liệu không thể được lưu trữ hoàn toàn trong bộ nhớ. Data generators có thể đọc dữ liệu từ đĩa, tiền xử lý và cung cấp dữ liệu cho mô hình theo từng batch. Keras cung cấp phương thức fit_generator để huấn luyện mô hình sử dụng data generators.
- **Data augmentation**: Keras cung cấp các tiện ích tích hợp sẵn để tăng kích thước của tập dữ liệu huấn luyện bằng cách áp dụng các biến đổi ngẫu nhiên cho dữ liệu hiện có. Data augmentation có thể giúp cải thiện khả năng tổng quát hóa của mô hình. Keras có nhiều chức năng tăng cường dữ liệu hình ảnh như xoay, dịch chuyển, thu phóng, lật, vv. Các chức năng này có thể dễ dàng áp dụng cho dữ liệu huấn luyện bằng cách sử dụng lớp ImageDataGenerator.
- **Model parallelism**: Keras hỗ trợ phân phối tính toán của một mô hình trên nhiều GPU hoặc nhiều máy tính. Điều này có thể được thực hiện bằng cách sử dụng hàm multi_gpu_model, hàm này lấy một mô hình và nhân bản nó trên nhiều GPU. Keras cũng hỗ trợ huấn luyện phân phối bằng cách sử dụng API tf.distribute.Strategy.

6. **Keras có hỗ trợ các hệ thống dữ liệu lớn không?**

Có, Keras hỗ trợ các bộ dữ liệu lớn. Keras cho phép sử dụng data generators để xử lý các bộ dữ liệu không thể lưu trữ hoàn toàn trong bộ nhớ. Bằng cách sử dụng data generators, mô hình có thể xử lý dữ liệu theo các batch nhỏ, giảm yêu cầu về bộ nhớ. Ngoài ra, Keras hỗ trợ huấn luyện phân phối trên nhiều GPU hoặc nhiều máy tính, cho phép huấn luyện hiệu quả trên các bộ dữ liệu lớn

7. **Keras có hoạt động tốt trên bộ dữ liệu lớn không?**

Keras được thiết kế để hoạt động tốt với các bộ dữ liệu lớn. Bằng cách sử dụng data generators, data augmentation và huấn luyện phân phối, Keras có thể xử lý các bộ dữ liệu lớn một cách hiệu quả. Khả năng huấn luyện mô hình với các bộ dữ liệu lớn là rất quan trọng đối với các nhiệm vụ học sâu, vì những mô hình này thường yêu cầu một lượng lớn dữ liệu để đạt được hiệu suất tốt. Mặc dù hiệu

suất cụ thể có thể phụ thuộc vào phần cứng và cơ sở hạ tầng sử dụng, Keras cung cấp các công cụ và kỹ thuật cần thiết để xử lý các bộ dữ liệu lớn một cách hiệu quả.

D. Nhóm câu hỏi liên quan đến Debugging trong Keras

8. **Có cách nào để debug khi gặp low-level errors trong Keras không?**
9. **Mình muốn biết về cách debug trong Keras**
10. **Khả năng debug model dùng Sequential trong Keras như thế nào?**

Về debugging trong Keras, bạn có tham khảo link sau đây: [Keras debugging tips](#).

Để tóm gọn lại ý, thì trong Keras, để debug một model, bạn có thể sử dụng một số mẹo như sau:

- Kiểm tra trên từng phần nhỏ code trước khi kiểm tra toàn bộ. Đây là một ý tưởng quan trọng, chúng ta nên đảm bảo từng phần nhỏ hoạt động đúng đắn trước khi kết hợp chúng với nhau.
- Keras cũng hỗ trợ các hàm `plot.summary()` và `plot_model()` cho phép chúng ra kiểm tra tính hợp lý về số lượng tham số cũng như kích thước cho mỗi layer.
- Để xem được quá trình thực hiện cập nhật trọng số của quá trình train cho một model. Keras cũng hỗ trợ cú pháp `run_eagerly=True` trong câu lệnh `fit()`.

Bên cạnh đó, Keras cũng hỗ trợ:

- **Hệ thống thông báo lỗi rõ ràng:** Khi một lỗi xuất hiện trong quá trình training, Keras sẽ cung cấp cho chúng ta một thông báo chi tiết về lỗi mà ta đang gặp phải.

Bạn có thể xem chi tiết trong bài viết sau: [keras blog](#)

- **Callbacks:** Là một function có thể được dùng để quản lý quá trình training, và thực hiện các tác vụ khi một điều kiện thỏa mãn. Ví dụ như "Early Stopping callback" sẽ dừng quá trình train khi mà độ loss đã giảm đến một mức nhất định, giảm thiểu các tính toán nhiều quá mức cần thiết.

Tóm lại, Keras hỗ trợ đa dạng các cách để có thể theo dõi, kiểm soát các lỗi phát sinh không mong muốn trong quá trình xây dựng mô hình, đối với các lỗi **low-level**, những lỗi liên quan đến tính toán dữ liệu, cấu trúc mô hình, bên cạnh những hỗ trợ của Keras, đòi hỏi người dùng còn phải thật sự hiểu rõ kiến trúc, cũng như chức năng của từng lớp, từng thành phần của mô hình mà bản thân đang xây dựng.

E. Nhóm câu hỏi liên quan đến sự khác nhau giữa các mô hình: Sequential Model vs Functional Model

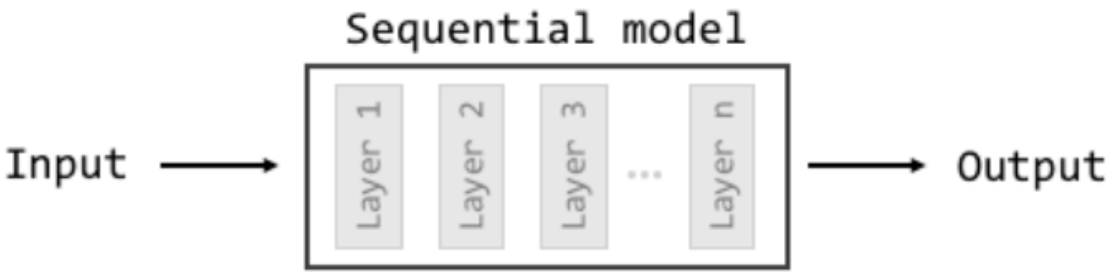
Nhóm nhận được 2 câu hỏi liên quan đến phân biệt mô hình Sequential API và Functional API, cụ thể như sau:

11. **Mình có một câu hỏi là sự khác biệt giữa Sequential API và Functional API trong Keras là gì? Nhóm có thể giải thích chi tiết hơn không?**
12. **Sequential API vs Functional API ?**

Vì nội dung như nhau nên chúng mình sẽ đưa ra một câu trả lời cho 2 câu hỏi.

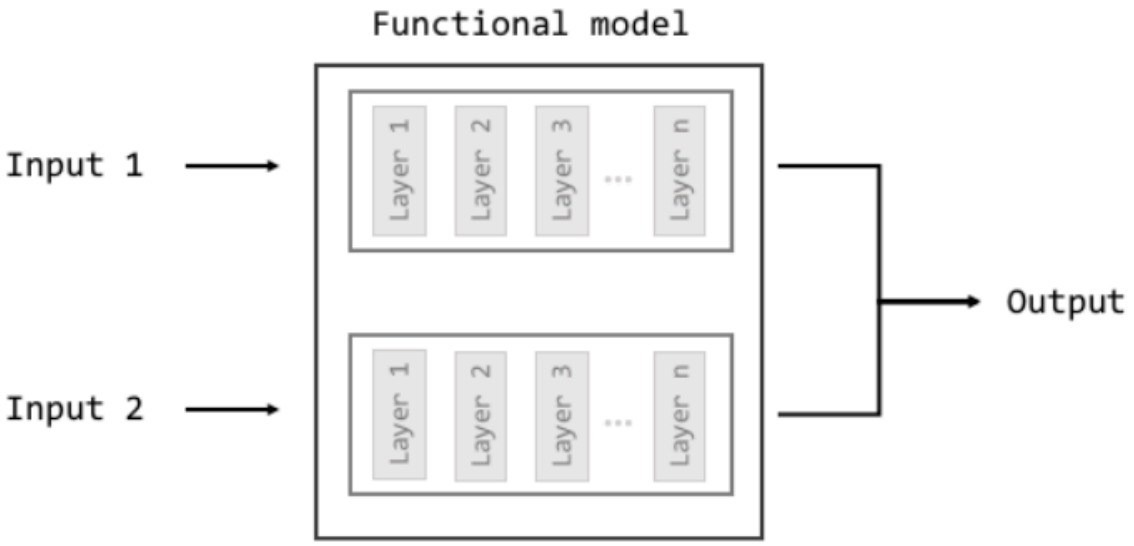
Trong bài thuyết trình, nhóm có trình bày định nghĩa của Sequential API và Functional API là:

- **Sequential API**: Mô hình mà trong đó mỗi lớp của nó có đúng 1 input và 1 output. Điều này có nghĩa là mỗi lớp nhận input từ 1 lớp trước đó, và output của nó chỉ truyền vào 1 lớp duy nhất.



Minh họa mô hình Sequential API
Source: Hello [Deep Learning] World - Brad Boehmke

- **Functional API:** Mô hình mà trong đó mỗi lớp của nó có thể có nhiều hơn 1 input và một output. Điều này có nghĩa là mỗi lớp của nó có thể nhận và xử lý input từ nhiều lớp, và nó có thể output ra dữ liệu để truyền vào nhiều lớp khác nhau.



Minh họa mô hình Functional API (Đây là một mô hình đơn giản.
Một mô hình phức tạp hơn có thể có hidden layer với nhiều input/output)
Source: Hello [Deep Learning] World - Brad Boehmke

Về cơ bản, sự khác nhau của các lớp là về phần cấu trúc. Nhưng để hiểu rõ được lý do tại sao lại có mô hình Sequential API và Functional API thì cần phải thường sẽ phù thuộc vào 3 lý do:

- **Đặc tính của dữ liệu**: Dữ liệu có nhiều dạng (xử lý dữ liệu trong một bảng có kiểu numeric và kiểu object), dữ liệu có thể phân ra nhiều dạng (ảnh được phân thành nhiều resolution khác nhau để học chiều sâu của ảnh)... Một bài toán cụ thể có thể nghĩ đến là Visual Question Answering (VQA) - khi chúng ta muốn train mô hình có thể đoán được nội dung của ảnh - chúng ta phải input thông tin của ảnh đó và miêu tả ảnh đó để train mô hình.
- **Đặc tính của mô hình**: Cấu trúc mô hình transformer (nội dung chi tiết về mô hình transformer nằm ngoài câu hỏi của mình nhưng nếu các bạn muốn tìm hiểu sâu hơn thì video này có thể giúp.)

F. Nhóm câu hỏi liên quan đến huấn luyện phân tán dữ liệu

13. Can we train LLM such as Llama Meta on a distributed dataset?

- Training LLM: Yes, we can train LLM with Keras. In fact, there is a session in Google I/O 2024 (just a while ago) that is dedicated to building LLM with Keras 3.0. Keras 3.0 also provides pretrained Llama which you can check out here.
- Training LLM on distributed dataset: Yes, we can train LLM (models) on distributed dataset. Keras supports distributed training with DataParallel and ModelParallel. Keras 3.0 takes advantage of its multiple backends, one of which is JAX - a data framework designed for high-performance numerical computing and large-scale machine learning, and there are pushes from the creators for developers to use Keras with JAX backend to train data on large scale, distributed systems. So Keras is definitely built having training on distributed systems in mind. You can read more about this here.

14. Keras có hỗ trợ cho việc huấn luyện phân tán (distributed training) không? Nếu có thì làm như thế nào?

- Keras có hỗ trợ cho việc huấn luyện phân tán. Cơ bản thì Keras sẽ tạo cho bạn một môi trường lập trình global (giống như làm việc trên một máy) - và sử dụng backend của nó (ex. JAX) để xử lý tensor và hệ thống phân tán ở dưới.
- Cụ thể, để làm việc này bạn có thể sử dụng lớp DataParallel and ModelParallel. DataParallel. Cách thức thực hiện huấn luyện phân tán mô hình nằm ngoài nội dung tìm hiểu của nhóm mình nên mình sẽ không đi sâu quá, nhưng bạn có thể tham khảo nguồn này để tìm hiểu kĩ hơn nhé.

G. Nhóm câu hỏi liên quan đến so sánh Keras với các framework khác

15. Trong các usecase nào việc sử dụng Keras thì vượt trội hơn so với các thư viện deep learning khác? Tại sao?

Keras vượt trội hơn so với các thư viện deep learning khác trong các usecase sau:

- **Prototyping**: Keras được thiết kế để có giao diện dễ sử dụng và dễ hiểu, giúp người dùng nhanh chóng xây dựng và kiểm tra các mô hình deep learning. Với Keras, người dùng có thể tạo ra một mô hình deep learning cơ bản chỉ trong vài dòng code. Điều này rất hữu ích khi cần thử nghiệm nhiều mô hình khác nhau để tìm hiểu và đánh giá hiệu suất của chúng.
- **Transfer learning**: Keras cung cấp nhiều mô hình deep learning đã được huấn luyện trước (pre-trained models) như VGG16, ResNet, và Inception. Việc sử dụng các mô hình này cho transfer learning rất dễ dàng trong Keras. Người dùng chỉ cần tải mô hình đã huấn luyện và thêm các lớp phù hợp vào cuối mô hình để thực hiện nhiệm vụ tùy chỉnh. Keras cung cấp các công cụ để tải và sử dụng các mô hình pre-trained một cách thuận tiện.

16. Tốc độ training khi train các mô hình deep learning giữa các thư viện với keras?

Tốc độ huấn luyện khi train các mô hình deep learning giữa các thư viện có thể khác nhau tùy thuộc vào nhiều yếu tố như cấu hình phần cứng, quy mô mô hình và cách triển khai. Tuy nhiên, Keras được xây dựng trên TensorFlow và có thể chạy trực tiếp trên TensorFlow, vì vậy hiệu năng của Keras khi train mô hình deep learning thường tương đương với TensorFlow.
Ngoài ra, Keras cũng có thể sử dụng các backend khác như Theano hoặc CNTK. Tuy nhiên, TensorFlow là backend chính được khuyến nghị và hỗ trợ tốt nhất trong Keras.
Còn so sánh với các mô hình của các thư viện khác thì chúng ta cần có bài toán cụ thể, cùng chạy trên phần cứng nào đó thì mới có kết quả so sánh chính xác giữa các mô hình deep learning trong cùng thư viện hoặc giữa các thư viên khác.

17. Tại sao tensorflow lại không thân thiện với beginner như Keras?

TensorFlow đối với người mới bắt đầu có thể khó tiếp cận hơn so với Keras bởi một số lý do:

1. **Độ phức tạp và cấp độ trừu tượng**

- **TensorFlow**: Là một framework mạnh mẽ và linh hoạt, TensorFlow cho phép kiểm soát chi tiết từng bước của quá trình xây dựng và huấn luyện mô hình. Điều này đồng nghĩa với việc người dùng cần phải viết nhiều mã hơn và hiểu rõ về các

khái niệm nền tảng như tensor, graph, session, v.v. Điều này có thể gây khó khăn cho người mới bắt đầu vì họ phải học nhiều khái niệm và cú pháp phức tạp.

- **Keras**: Được thiết kế với mục tiêu làm cho việc xây dựng và huấn luyện mô hình học sâu trở nên đơn giản và dễ tiếp cận hơn. Keras cung cấp một API ở mức độ cao, dễ đọc và dễ sử dụng. Các thao tác phức tạp được ẩn giấu bên dưới, giúp người dùng tập trung vào thiết kế và huấn luyện mô hình mà không cần lo lắng về các chi tiết kỹ thuật

2. **Cú pháp và API**

- **TensorFlow**: API của TensorFlow thường phức tạp hơn và yêu cầu người dùng phải viết nhiều mã để thực hiện các tác vụ đơn giản. Việc này có thể làm người mới bắt đầu cảm thấy choáng ngợp.
- **Keras**: Cung cấp một API trực quan và dễ hiểu, với cú pháp gọn gàng và logic. Việc tạo và huấn luyện một mô hình trong Keras thường chỉ cần vài dòng mã, giúp người mới bắt đầu dễ dàng nắm bắt và sử dụng.

3. **Mục tiêu sử dụng**

- **TensorFlow**: Thích hợp cho các dự án lớn, phức tạp và yêu cầu tùy chỉnh cao. Nó thường được sử dụng trong nghiên cứu và phát triển các mô hình học sâu tiên tiến.
- **Keras**: Thích hợp cho việc prototyping nhanh, các dự án nhỏ hơn hoặc khi người dùng muốn nhanh chóng triển khai và thử nghiệm các ý tưởng mới.

H. Nhóm câu hỏi liên quan đến tiền xử lý dữ liệu với Keras

18. Keras có khả năng tạo data augmentation tốt không?

19. Việc xử lý các điểm dữ liệu ở góc, cạnh của ảnh sẽ được xử lý như thế nào?

20. Làm thế nào để xử lý dữ liệu thời gian (time series) bằng Keras?

21. Nếu chúng ta chỉ sử dụng ML thì để nâng cao hiệu suất sẽ cần thêm những tiền xử lý hoặc thêm những biến hỗ trợ. Liệu deep learning có cần nâng cao hiệu suất bằng cách tương tự hay không?

Câu 18:

- Đầu tiên, **Data Augmentation** là kỹ thuật giúp chúng ta gia tăng kích thước cũng như sự đa dạng của tập dữ liệu huấn luyện bằng cách tạo ra các biến thể ngẫu nhiên từ các dữ liệu gốc, giúp cải thiện hiệu quả và độ chính xác của mô hình.
- Một số lợi ích của Data Augmentation có thể kể đến như sau:
    + **Giảm thiểu overfitting:** Nhiều biến thể khác nhau sẽ giúp mô hình không bị quá khớp với dữ liệu huấn luyện
    + **Tăng hiệu quả học:** Nhiều dữ liệu hơn, mô hình có thể học được nhiều đặc trưng hơn => Hiệu quả huấn luyện tăng

- Tuy nhiên, Data Augmentation cũng có thể đem lại những điểm bất lợi sau:
    + Những biến thể không mong muốn: Khi dữ liệu được biến đổi đến mức ảnh hưởng đến hình dạng cũng như ý nghĩa của ảnh,
    + Tăng chi phí tính toán: Việc tạo ra thêm nhiều phiên bản dữ liệu đồng nghĩa với việc quá trình tính toán, quản lý cũng sẽ phức tạp hơn.
    + Khó kiểm soát chất lượng dữ liệu
    + Không phù hợp với mọi đối tượng dữ liệu.

- Keras cung cấp các lớp liên quan đến tiền xử lý để thực hiện data augmentation cho hình ảnh, cũng như, mỗi lớp đều sẽ có các tham số cấu hình khác nhau, hỗ trợ người dùng trong quá trình điều chỉnh công việc của mình.

Thế nhưng, việc điều chỉnh các tham số của các hàm trên lại phụ thuộc vào chính bản thân chúng ta, nếu các tham số không hợp lý, thì khả năng tạo ra các kết quả dữ liệu không mong muốn rất cao.

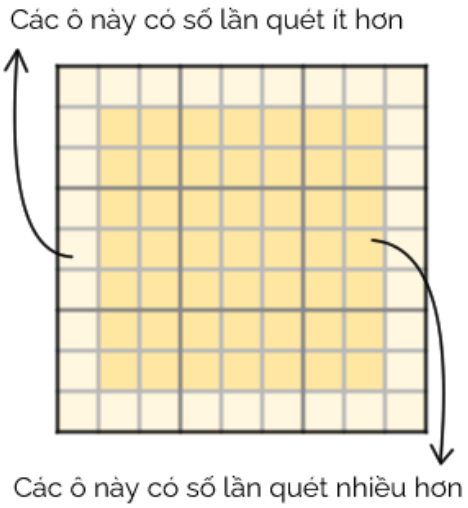
Như vậy, tổng hợp lại, chất lượng của data augmentation có tốt hay không, còn phụ thuộc vào cách chúng ta quan sát, hiểu và đánh giá dữ liệu, việc điều chỉnh hay lựa chọn phép biến đổi nào phụ thuộc lớn vào bài toán cũng như các yêu cầu về chất lượng được đặt ra.

Câu 19:

- Có vẻ câu hỏi của bạn đang đề cập đến quá trình thực hiện phép tích chập của lớp **Convolutional** trong Keras. Mình xin đính chính là ví dụ các bạn được xem

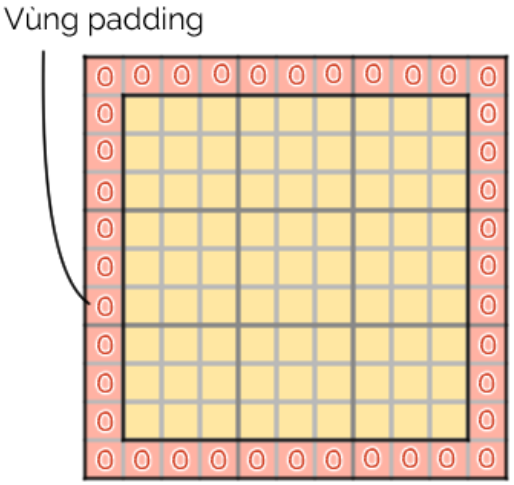
trong bài trình bày của nhóm mình là phiên bản đơn giản của phép tích chập. Do đó sẽ chưa khái quát được hết các trường hợp.

- Về vấn đề góc và cạnh:



- Các điểm ở góc và cạnh, nếu áp dụng cách tích chập thông thường, thì những điểm ở góc hay cạnh (các ô nhạt màu) chỉ được thực hiện tích một hoặc một vài lần, trong khi đó, điểm ở trung tâm ảnh (điểm màu đậm), sẽ được thực hiện tích chập rất nhiều lần, điều này có thể dẫn đến sự mất mát thông tin trên các điểm ở góc và cạnh.

Để giải quyết vấn đề này, trong quá trình tích chập, sẽ có một thuật ngữ là **"padding"**, đề cập đến việc thêm các giá trị 0 xung quanh ảnh gốc của chúng ta, và thực hiện tích chập trên phiên bản mở rộng của ảnh. Khi đó, các điểm vốn là góc và cạnh sẽ được dời vào trong, đồng nghĩa với việc thông tin mất mát của chúng sẽ bị giảm thiểu đi



Tuy vậy, cô thêm vào bao nhiêu là hợp lý? Mặc dù có thể tính toán thủ công, nhưng với Keras, việc tính toán sẽ được thực hiện tự động trong lớp Conv2D dựa trên tham số **padding**, với 2 lựa chọn là:

- **Valid:** Không có padding, kích thước của đầu ra sẽ giảm so với kích thước gốc, tát nhiên là kết quả trên góc và cạnh cũng sẽ không được như mong muốn.
- **Same:** Thực hiện padding sao cho kết quả đầu ra sau khi tích chập phải có cùng kích thước với ảnh đầu vào.

Câu 20: Làm thế nào để xử lý dữ liệu thời gian (time series) bằng Keras?

- Câu hỏi của bạn làm mình nghĩ theo 2 hướng:
    + Keras có hỗ trợ tiền xử lý dữ liệu time-series hay không?
    + Keras hỗ trợ mô hình nào trong việc phân tích dữ liệu time-series

- Mục tiêu của Keras là xây dựng một mô hình Neural Network, do đó các quá trình tiền xử lý dữ liệu time-series như chuyển đổi kiểu dữ liệu, tính toán, xử lý nhiễu nên được thực hiện độc lập thông qua các thư viện khác như Numpy hay Pandas, vốn đã quen thuộc với chúng ta, Keras sẽ nhận giá trị đầu vào cho mô hình và thực hiện tác vụ mong muốn.
- Mặc dù vậy, Keras vẫn hỗ trợ rất nhiều hàm thao tác cơ bản trên kiểu dữ liệu **tensor**: Ops API. tức là về lý thuyết, bạn có thể thực hiện việc chuyển đổi dữ liệu về kiểu tensor và thực hiện các thao tác trên đó, tuy nhiên, số lượng thao tác hỗ trợ cũng hạn chế và không được phong phú như Pandas hay Numpy
- Nói về các mô hình có thể xem xét cho việc phân tích time-series. Keras cho phép chúng ta cài đặt rất nhiều mô hình hỗ trợ cho time-series như LMTS, GRU,...
- Bạn có thể xem nhiều ví dụ hơn ở đây: Timeseries

Câu 21:

Như chúng ta đã biết, chất lượng dữ liệu đầu vào rất quan trọng, cho bất kì mô hình nào, cho nên dù là Machine Learning hay Deep Learning, thì một bộ dữ liệu tốt luôn đem lại hiệu quả tốt hơn.

Một ví dụ mình có thể đưa ra cho bạn ngay trong phần demo của nhóm mình: demo
Ở phần cuối cùng, mình đã áp dụng một phép biến đổi, đưa dữ liệu về dạng chuẩn và dễ học hơn, và kết quả của mô hình cũng đã có sự cải thiện so với bộ dữ liệu gốc ban đầu,

I. Nhóm câu hỏi liên quan đến Demo

22. Sự khác biệt cụ thể giữa tập test và tập validation trong demo?

Tập test và tập validation có ý nghĩa khác nhau. Về cơ bản tập validation được sử dụng để đánh giá độ hiệu quả (accuracy) của quá trình train model trước khi test lại bằng tập test. Để hiểu hơn về nguyên nhân tại sao chúng ta lại sử dụng tập validation và test, hay sử dụng một phép ẩn dụ. Tưởng tượng bạn học để đi thi THPT quốc gia - giống như một mô hình sẽ được train để test. Trước khi đi thi, bạn có một số đề ôn thi có giải. Bạn có thể tưởng tượng bạn học và giải một số đề thi, sau đó sử dụng một số đề để đánh giá lại khả năng của mình. Điều này giống như một mô hình được train (học) và test bằng tập validation (đánh giá khả năng) vậy. Việc sử dụng tập validation tách từ tập train giúp chúng ta đánh giá khả năng của mô hình và quan trọng hơn là nhận ra các nhược điểm cũng như khắc phục nó trước khi test.

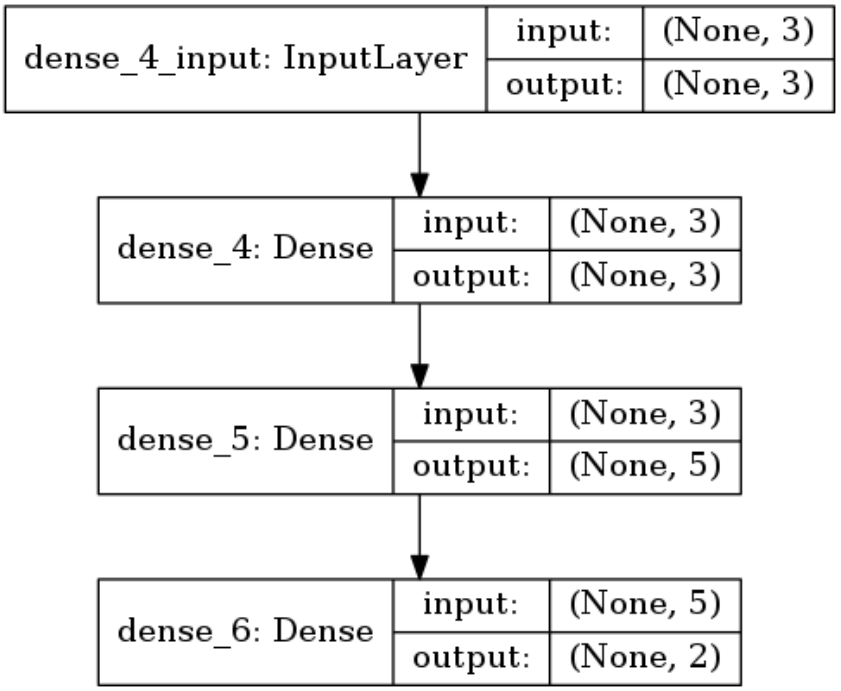
Về phần demo, với mục tiêu là tạo ra một mô hình đơn giản, không hướng đến hiệu suất hay kết quả tối ưu, cho nên việc mình đã không sử dụng tập validation, mà thay vào đó là dùng trực tiếp tập test như là tập validation luôn.

Nếu bạn muốn đọc thêm về chủ đề này, bạn có thể xem thêm ở tài liệu này nhé.

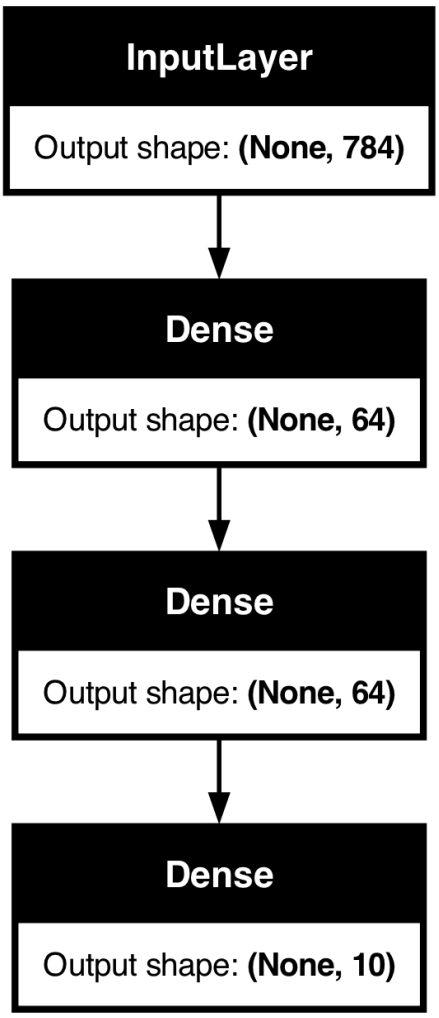
K. Các câu hỏi khác

23. Keras có hỗ trợ visualize model hay không?

- Keras có hỗ trợ visualize model qua phương thức plot_model và model_to_dot. Điểm khác nhau lớn nhất của 2 phương thức này chính là plot_model lưu một file hình ảnh ('png') minh họa của mô hình, model_to_dot thì không. Cụ thể bạn có thể đọc thêm ở đây.



Ví dụ sử dụng phương thức model_to_dot để visualize model
Source: How to plot Keras models - Zhang Yang



Ví dụ sử dụng phương thức plot_model để visualize model
Source: The functional API - Keras 3.0 documentation

24. **Can Keras integrate with other programming languages?**

- I'm not sure what you mean by 'integrate with other programming languages', but if you mean whether you can use Keras in another programming language, the answer is yes. There are Keras interfaces for other programming languages - namely R (you can check it out here). You can also load Keras models with other language environments - like JS and Swift - because Keras can utilise the Tensor Flow integration to multiple platforms (Web, mobile, cloud) for deployment.

25. **Công việc của lớp "Flatten" trong Keras là gì?**

- Mục tiêu chính của chúng ta khi sử dụng lớp "Flatten" chính là biến đổi dữ liệu đầu vào, từ đa chiều thành một chiều.
- "Flatten" không thay đổi số lượng mẫu trong batch mà chỉ thay đổi hình dạng của mỗi mẫu.

- Đối với bài toán **CNN**: Flatten là một lớp cần thiết, trung gian giữa lớp **Convolutional,** nơi trích xuất ra các đặc trưng, và lớp **Dense**, nơi thực hiện thao tác tính toán và phân loại dựa trên những đặc trưng đó. Cụ thể:
    + Sau khi qua các lớp Conv2D và MaxPooling2D, đầu ra của mô hình sẽ có hình dạng (batch_size, height, width, channels).
    + Lớp Flatten chuyển đổi ma trận đầu ra cuối cùng , mang kích thước (batch_size, height, width, channels) thành một vector 1D với hình dạng (batch_size, height * width * channels).
    + Vector 1D này sau đó được đưa vào các lớp Dense để thực hiện nhiệm vụ phân loại.

conv2d_1 (Conv2D)	(None, 14, 14, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 64)	0
dropout (Dropout)	(None, 7, 7, 64)	0
flatten (Flatten)	(None, 3136)	0

Với hình trên, sau lớp max_pooling_2d_1, kích thước của đầu ra là (None, 7, 7, 64), lớp Flatten sẽ tiến hành đưa ma trận đó về dạng 1D, với kích thước là (None, 7*7*64) = (None, 3136).
Có lẽ bạn sẽ thắc mắc None ở đây là gì, đó là cách thể hiện của số lượng ảnh trong một batch dữ liệu của chúng ta, khi vừa khởi tạo, giá trị batch chưa được xác định cụ thể, nên xuất hiện None. Như vậy, nếu xét trên từng ảnh riêng biệt, thì sau khi qua lớp Flatten, một ảnh sẽ được chuyển từ kích thước 3d (7,7,64) sang một mảng 1d với độ dài là 3136.