

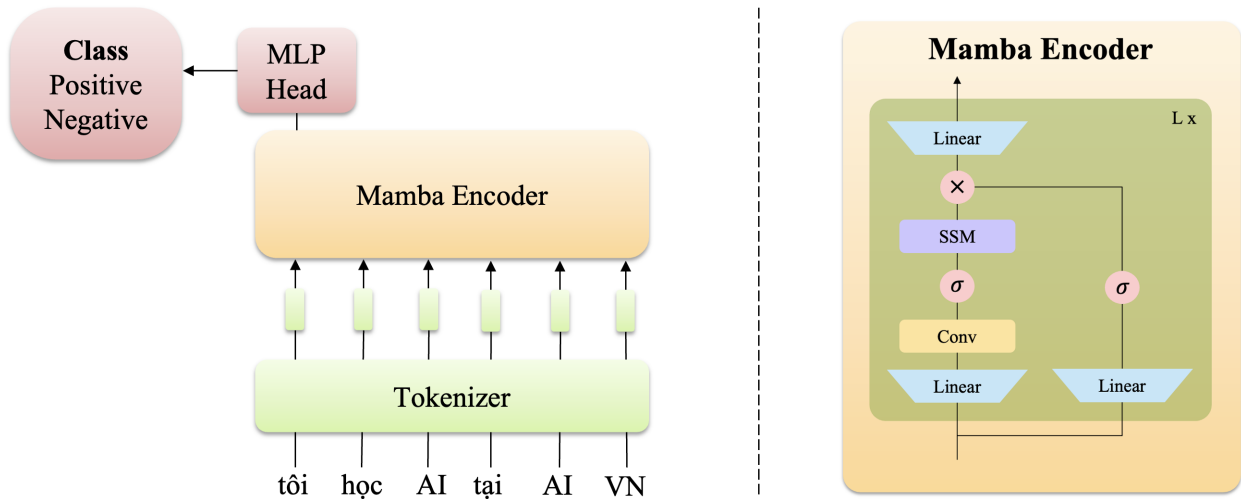
Text Classification with Mamba - Project

Minh-Duc Bui, Khai-Xuan Trinh, và Quang-Vinh Dinh

Ngày 3 tháng 3 năm 2024

Phần I: Giới thiệu

Gần đây, **Mamba** là kiến trúc mới ra mắt và được sự hưởng ứng mạnh mẽ từ cộng đồng các nhà nghiên cứu. Mamba trở thành trend vì khả năng vượt trội hơn Transformer (kiến trúc phổ biến ở thời điểm hiện tại). Sự vượt trội được thể hiện ở cả 3 tiêu chí chính để đánh giá 1 model: accuracy, speed, và computational cost.



Trong project này, ta sẽ tìm hiểu cơ bản về kiến trúc Mamba và áp dụng Mamba vào bài toán text classification.

Phần II: Nội dung

1. Mô tả dataset IMDB

IMDB dataset là bộ data bao gồm 50,000 đánh giá về phim. Đây là bộ dữ liệu được sử dụng cho việc phân loại đánh giá tiêu cực và tích cực. Bộ dữ liệu được chia làm 2 phần bằng nhau, 25,000 mẫu để train, và 25,000 mẫu để kiểm thử. Bên cạnh đó, bộ dữ liệu cũng cung cấp 50,000 mẫu dữ liệu chưa đánh nhãn để hỗ trợ quá trình train. Tuy nhiên trong project này ta chỉ sử dụng phần dữ liệu đã được đánh nhãn để train model.

Review: Previous reviewer Claudio Carvalho gave a much better recap of the film's plot details than I could. What I recall mostly is that it was just so beautiful, in every sense - emotionally, visually, editorially - just gorgeous.

If you like movies that are wonderful to look at, and also have emotional content to which that beauty is relevant, I think you will be glad to have seen this extraordinary and unusual work of art.

On a scale of 1 to 10, I'd give it about an 8.75. The only reason I shy away from 9 is that it is a mood piece. If you are in the mood for a really artistic, very romantic film, then it's a 10. I definitely think it's a must-see, but none of us can be in that mood all the time, so, overall, 8.75.

Label: Positive

Review: Technically I'm a Van Damme Fan, or I was. this movie is so bad that I hated myself for wasting those 90 minutes. Do not let the name Isaac Florentine (Undisputed II) fool you, I had big hopes for this one, depending on what I saw in (Undisputed II), man.. was I wrong ??! all action fans wanted a big comeback for the classic action hero, but i guess we wont be able to see that soon, as our hero keep coming with those (going -to-a-border - far-away-town-and -kill -the-bad-guys- than-comeback- home) movies I mean for God's sake, we are in 2008, and they insist on doing those disappointing movies on every level. Why ?!!!! Do your self a favor, skip it.. seriously.

Label: Negative

Hình 1: Ví dụ minh họa về dataset IMDB.

2. Model Mamba cho bài toán Text classification

- (a) **Install and import libraries:** Đầu tiên ta sẽ install một số thư viện cần thiết của Huggingface và Mamba:

```
1 !pip install datasets evaluate accelerate
2 !pip install causal-conv1d>=1.1.0
3 !pip install mamba-ssm
```

Sau đó ta sẽ tiến hành login vào HuggingFace để download dataset và model có sẵn. Khi chạy block code này thì HuggingFace sẽ đưa ra một đường dẫn đến trang [HuggingFace](#) để lấy mã token. Lưu ý để thuận tiện cho quá trình train và đưa model lên Huggingface Hub thì ta nên sử dụng token có quyền ghi của Huggingface.

```
1 from huggingface_hub import notebook_login
2 notebook_login()
```

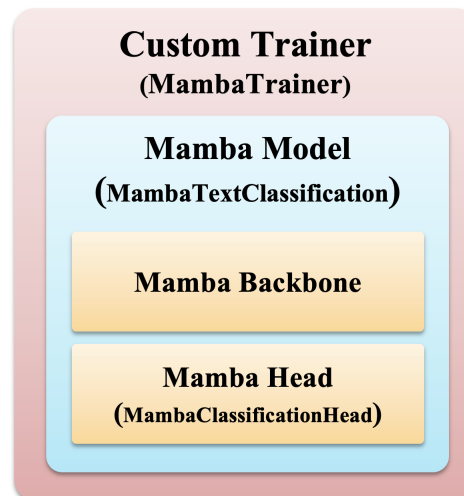
Cuối cùng ta sẽ import các thư viện chính được sử dụng trong phần này:

```
1 import os
2 import random
3 import json
4 import torch
5 import torch.nn as nn
6 from collections import namedtuple
7 from dataclasses import dataclass, field, asdict
8 from mamba_ssm.models.mixer_seq_simple import MambaLMHeadModel
9 from mamba_ssm.utils.hf import load_config_hf, load_state_dict_hf
10
11 import evaluate
12 import numpy as np
13 from datasets import load_dataset
14 from transformers import Trainer
15 from transformers import AutoTokenizer, TrainingArguments
```

(b) **Download dataset:**

```
1 # Tải bộ dataset
2 imdb = load_dataset("imdb")
```

(c) **Build Custom Mamba Model:** Xây dựng model Mamba để phân loại văn bản.



- Setup config:

```

1 # Config class của Mamba
2 class MambaConfig:
3     d_model: int = 2560
4     n_layer: int = 64
5     vocab_size: int = 50277
6     ssm_cfg: dict = field(default_factory=dict)
7     rms_norm: bool = True
8     residual_in_fp32: bool = True
9     fused_add_norm: bool = True
10    pad_vocab_size_multiple: int = 8
11
12    def to_json_string(self):
13        return json.dumps(asdict(self))
14
15    def to_dict(self):
16        return asdict(self)

```

- Định nghĩa class head (classifier) để phục vụ cho việc phân loại văn bản:

```

1 # Định nghĩa class head để phân loại
2 class MambaClassificationHead(nn.Module):
3     def __init__(self, d_model, num_classes, **kwargs):
4         super(MambaClassificationHead, self).__init__()
5         # Sử dụng một lớp tuyến tính để thực hiện phân loại dựa trên
6         # đầu vào có kích thước d_model và num_classes cần phân loại.
7         self.classification_head = nn.Linear(d_model, num_classes,
8         **kwargs)
9
10    def forward(self, hidden_states):
11        return self.classification_head(hidden_states)

```

- Định nghĩa model Mamba:

```

1 class MambaTextClassification(MambaLMHeadModel):
2     def __init__(
3         self,
4         config: MambaConfig,
5         initializer_cfg=None,
6         device=None,
7         dtype=None,
8     ) -> None:
9         super().__init__(config, initializer_cfg, device, dtype)
10
11        # Tạo một đầu phân loại sử dụng MambaClassificationHead với
12        # kích thước đầu vào là d_model và số lớp là 2.
13        self.classification_head = MambaClassificationHead(d_model=
14        config.d_model, num_classes=2)
15
16        del self.lm_head
17
18    def forward(self, input_ids, attention_mask=None, labels=None):

```

```

17         # Truyền input_ids qua model gốc để nhận hidden_states.
18         hidden_states = self.backbone(input_ids)
19
20         # Lấy trung bình của hidden_states theo chiều thứ 2 để tạo
21         ra [CLS] feature đại diện
22         mean_hidden_states = hidden_states.mean(dim=1)
23
24         # Đưa mean_hidden_states qua đầu phân loại để nhận logits.
25         logits = self.classification_head(mean_hidden_states)
26
27         if labels is None:
28             ClassificationOutput = namedtuple("ClassificationOutput",
29 ["logits"])
30             return ClassificationOutput(logits=logits)
31         else:
32             ClassificationOutput = namedtuple("ClassificationOutput",
33 ["loss", "logits"])
34             loss_fct = nn.CrossEntropyLoss()
35             loss = loss_fct(logits, labels)
36
37             return ClassificationOutput(loss=loss, logits=logits)
38
39     def predict(self, text, tokenizer, id2label=None):
40         input_ids = torch.tensor(tokenizer(text)['input_ids'],
41 device='cuda')[None]
42         with torch.no_grad():
43             logits = self.forward(input_ids).logits[0]
44             label = np.argmax(logits.cpu().numpy())
45
46         if id2label is not None:
47             return id2label[label]
48         else:
49             return label
50
51     @classmethod
52     def from_pretrained(cls, pretrained_model_name, device=None,
53 dtype=None, **kwargs):
54         # Tải cấu hình từ model đã được train trước đó.
55         config_data = load_config_hf(pretrained_model_name)
56         config = MambaConfig(**config_data)
57
58         # Khởi tạo model từ cấu hình và chuyển nó đến thiết bị và ki
59         ểu dữ liệu mong muốn.
60         model = cls(config, device=device, dtype=dtype, **kwargs)
61
62         # Tải trạng thái model đã được train trước đó.
63         model_state_dict = load_state_dict_hf(pretrained_model_name,
64 device=device, dtype=dtype)
65         model.load_state_dict(model_state_dict, strict=False)
66
67         # In ra các tham số embedding mới được khởi tạo.
68         print("Newly initialized embedding:", set(model.state_dict()
69 .keys()) - set(model_state_dict.keys()))
70         return model

```

- Cuối cùng ta sẽ tải trọng số và tokenizer của model Mamba đã được pretrain từ trước.

Trọng số của model Mamba pretrain sẽ không bao gồm các tham số của phần head (classifier) MambaClassificationHead mà ta tự định nghĩa. Do đó, phần head này sẽ được khởi tạo tham số từ đầu:

```

1 # Tải model Mamba từ model đã được train trước đó.
2 model = MambaTextClassification.from_pretrained("state-spaces/mamba
    -130m")
3 model.to("cuda")
4
5 # Tải tokenizer của model Mamba từ model gpt-neox-20b.
6 tokenizer = AutoTokenizer.from_pretrained("EleutherAI/gpt-neox-20b")
7 # Đặt id của token pad bằng id của token eos trong tokenizer.
8 tokenizer.pad_token_id = tokenizer.eos_token_id

```

- (d) **Preprocess dataset:** Trong phần này ta sẽ tiến hành tokenize dataset cho tập train và tập test. Vì số lượng sample của tập test khá lớn nên để thuận tiện cho quá trình train ta sẽ lấy ra 1 phần nhỏ của tập test để đánh giá model.

```

1 # Tạo chức năng tiền xử lý để mã hóa văn bản và cắt bớt các chuỗi không
    dài hơn độ dài đầu vào tối đa của mã thông báo
2 def preprocess_function(examples):
3     samples = tokenizer(examples["text"], truncation=True)
4     # Không cần attention_mask
5     # Cụ thể hơn về token masking của mamba có thể tham khảo: https://
    github.com/state-spaces/mamba/issues/49
6     samples.pop('attention_mask')
7     return samples
8
9 # Thực hiện mã hóa văn bản
10 tokenized_imdb = imdb.map(preprocess_function, batched=True)
11
12 # Set seed cho hàm random
13 random.seed(42)
14
15 # Tạo tập train và test
16 train_dataset = tokenized_imdb["train"]
17 test_dataset = tokenized_imdb["test"]
18
19 # Tạo tập evaluation để đánh giá trong lúc train
20 # Do số lượng tập test lớn nên chỉ lấy mẫu 1% tập dữ liệu test để đánh
    giá
21 total_samples = len(test_dataset)
22 eval_samples = int(0.1 * total_samples)
23 eval_indices = random.sample(range(total_samples), eval_samples)
24 eval_dataset = test_dataset.select(eval_indices)

```

- (e) **Evaluation metric:** Để đánh giá performance của model ta sẽ sử dụng metric accuracy từ thư viện evaluate:

```

1 # Tải module "accuracy" từ thư viện evaluate.
2 accuracy = evaluate.load("accuracy")
3

```

```

4 # Định nghĩa hàm compute_metrics để tính các độ đo hiệu suất (metrics)
  cho việc đánh giá model.
5 def compute_metrics(eval_pred):
6     predictions, labels = eval_pred
7
8     # Lấy chỉ số của lớp có xác suất cao nhất trong predictions.
9     predictions = np.argmax(predictions, axis=1)
10
11    # Sử dụng module "accuracy" để tính độ chính xác dựa trên
    predictions và labels.
12    return accuracy.compute(predictions=predictions, references=labels)

```

(f) **Train model:** Sau khi đã chuẩn bị xong dataset, ta sẽ tiến hành setup một số tham số trong quá trình train và tiến hành train model.

- Trước hết, ta sẽ định nghĩa một số hyper-parameter mà ta sẽ sử dụng để train model:

```

1 # Định nghĩa tên project để log thông tin quá trình train trên wandb
2 # os.environ["WANDB_PROJECT"] = "mamba_tutorial"
3
4 # Định nghĩa các tham số train trong class TrainingArguments.
5 # Cụ thể hơn về các tham số hỗ trợ có thể tham khảo: https://
  huggingface.co/docs/transformers/main_classes/trainer
6 training_args = TrainingArguments(
7     output_dir="mamba_text_classification", # Tên folder output
8     learning_rate=5e-5,
9     per_device_train_batch_size=4, # Số lượng train sample trên mỗi
    device
10    per_device_eval_batch_size=16, # Số lượng eval sample trên mỗi
    device
11    num_train_epochs=1, # Số epoch train
12    warmup_ratio=0.01, # Tỷ lệ tăng dần lr trong giai đoạn warmup
13    lr_scheduler_type="cosine", # Loại scheduler để giảm lr
14    report_to="none", # "wandb" nếu muốn log kết quả
15    evaluation_strategy="steps", # Xác định metric đánh giá sau mỗi
    số bước
16    eval_steps=0.1, # Số bước giữa các đợt đánh giá
17    save_strategy="steps", # Xác định khi nào lưu checkpoint
18    save_steps=0.1, # Số bước giữa các lần lưu checkpoint
19    logging_strategy="steps", # Xác định khi nào in thông tin log
20    logging_steps=1, # Số bước giữa các lần in thông tin log
21    push_to_hub=True, # Đẩy kết quả lên Hub
22    load_best_model_at_end=True, # Load model có kết quả evaluation
    tốt nhất trong quá trình train
23 )

```

- Sau đó ta sẽ khởi tạo class MambaTrainer kế thừa từ class Trainer. Đầu tiên, ta sẽ tạo hàm compute_loss() để định nghĩa hàm loss sử dụng trong quá trình train. Vì ta đã triển khai hàm loss là cross-entropy trong hàm forward của model, nên ta chỉ cần trích xuất giá trị mất mát từ kết quả trả về của hàm forward. Sau đó ta sẽ tiếp tục code hàm save_model() để định nghĩa cách lưu model. Để lưu model, ta cần ghi lại các tham số, tokenizer, và cấu hình (config) của model.

```

1 # Định nghĩa một class MambaTrainer kế thừa từ class Trainer.
2 class MambaTrainer(Trainer):
3
4     # Định nghĩa hàm compute_loss để tính toán hàm mất mát trong quá
      trình train.
5     def compute_loss(self, model, inputs, return_outputs=False):
6         # Lấy giá trị input_ids và labels từ inputs.
7         input_ids = inputs.pop("input_ids")
8         labels = inputs.pop('labels')
9
10        # Gọi hàm forward của model với input_ids và labels để nhận
      các kết quả.
11        outputs = model(input_ids=input_ids, labels=labels)
12
13        # Lấy giá trị loss từ kết quả của model.
14        loss = outputs.loss
15
16        # Trả về cả loss và outputs nếu return_outputs là True, ngượ
      c lại chỉ trả về loss.
17        return (loss, outputs) if return_outputs else loss
18
19    # Định nghĩa hàm save_model để lưu model trong quá trình train.
20    def save_model(self, output_dir = None, _internal_call = False):
21        # Kiểm tra nếu thư mục lưu trữ không được chỉ định, sử dụng
      thư mục mặc định từ đối số 'args'.
22        if output_dir is None:
23            output_dir = self.args.output_dir
24
25        # Nếu thư mục đầu ra không tồn tại, tạo mới nó.
26        if not os.path.exists(output_dir):
27            os.makedirs(output_dir)
28
29        # Lưu trạng thái của model PyTorch vào file 'pytorch_model.
      bin' trong thư mục đầu ra.
30        torch.save(self.model.state_dict(), f"{output_dir}/
      pytorch_model.bin")
31
32        # Lưu trạng thái của tokenizer vào thư mục đầu ra.
33        self.tokenizer.save_pretrained(output_dir)
34
35        # Lưu cấu hình của model vào file 'config.json' trong thư mụ
      c đầu ra.
36        with open(f'{output_dir}/config.json', 'w') as f:
37            json.dump(self.model.config.to_dict(), f)

```

- Cuối cùng ta sẽ khởi tạo class MambaTrainer, đây là class chính để train model. Sau khi đã khởi tạo thì ta chỉ cần gọi trainer.train() thì quá trình train model sẽ được tiến hành:

```

1 # Khởi tạo class MambaTrainer để thực hiện quá trình train của
      model.
2 trainer = MambaTrainer(
3     model=model, # Model cần train
4     train_dataset=train_dataset, # Dữ liệu train
5     eval_dataset=eval_dataset, # Dữ liệu đánh giá

```



```

6     tokenizer=tokenizer, # Tokenizer sử dụng để mã hóa dữ liệu
7     args=training_args, # Các tham số train đã được định nghĩa trước
      đó
8     compute_metrics=compute_metrics # Hàm tính các độ đo hiệu suất (
      metrics) cho đánh giá
9     )
10 # Bắt đầu quá trình train bằng cách gọi hàm train() trên classs
      trainer.
11 trainer.train()

```

[6250/6250 1:43:44, Epoch 1/1]

Step	Training Loss	Validation Loss	Accuracy
625	0.006700	0.310832	0.915800
1250	1.815400	0.234681	0.931000
1875	0.034100	0.299548	0.924600
2500	0.005200	0.231175	0.936400
3125	0.000100	0.263437	0.932000
3750	0.001600	0.224985	0.942000
4375	1.891200	0.211221	0.945200
5000	0.002500	0.219075	0.946000
5625	0.014900	0.212537	0.945000
6250	0.003500	0.211162	0.945400

- Sau khi quá trình train hoàn tất, ta sẽ đưa weight, config của model lên HuggingFace Hub để lưu lại:

```

1 # Đẩy model lên huggingface hub
2 trainer.push_to_hub(commit_message="Training complete")
3
4 >> Output: CommitInfo(commit_url='https://huggingface.co/
      trinxuankhai/mamba_text_classification/commit/816827
      ae91a91dd9006a9ef66ecef837382998b', commit_message='Training
      complete', commit_description='', oid='816827
      ae91a91dd9006a9ef66ecef837382998b', pr_url=None, pr_revision=
      None, pr_num=None)

```

- (g) **Run Testing:** Sau khi đã hoàn tất quá trình train, ta sẽ đánh giá model trên tập test và in ra kết quả đánh giá của model:

```

1 # Thực hiện dự đoán trên tập dữ liệu validation
2 outputs = trainer.predict(test_dataset)
3 print(outputs.metrics)
4
5 >> Output: {'test_loss': 0.21128389239311218, 'test_accuracy': 0.94708,
      'test_runtime': 1308.2019, 'test_samples_per_second': 19.11, '
      test_steps_per_second': 1.195}

```

- (h) **Load and inference model from Hub:** Ở phần trước, sau khi ta đưa model lên Hugging-face Hub, nếu muốn inference model ta có thể gọi hàm `from_pretrained` của model Mamba ta đã định nghĩa ở trước để load pretrained model. Sau đó ta sẽ truyền văn bản cần phân loại, tokenize và id của từng class vào hàm `predict` của model để thực hiện dự đoán kết quả.

```
1 # Tải model Mamba từ model đã được train trước đó.
2 model = MambaTextClassification.from_pretrained("trinhxuankhai/
    mamba_text_classification")
3 model.to("cuda")
4
5 # Tải tokenizer của model Mamba từ model đã được train trước đó.
6 tokenizer = AutoTokenizer.from_pretrained("trinhxuankhai/
    mamba_text_classification")
7 # Đặt id của token pad bằng id của token eos trong tokenizer.
8 tokenizer.pad_token_id = tokenizer.eos_token_id
```

Sau đây ta sẽ chạy thử một sample trên tập test:

```
1 id2label = {0: "NEGATIVE", 1: "POSITIVE"}
2 text = imdb['test'][0]['text']
3 label = imdb['test'][0]['label']
4 response = model.predict(text, tokenizer, id2label)
5 print(f'Classify: {text}\nGT: {id2label[label]}\nPredict: {response}')
6
7 >> Output:
8 - Classify: I love sci-fi and am willing to put up with a lot. Sci-fi
    movies/TV are usually underfunded, under-appreciated and
    misunderstood. I tried to like this, I really did, but it is to good
    TV sci-fi as Babylon 5 is to Star Trek (the original).
9 - GT: NEGATIVE
10 - Predict: NEGATIVE
```

Phần III: Câu hỏi trắc nghiệm

1. Trong state space model, dạng recurrent phù hợp cho quá trình inference vì?
 - (a) Khả năng tính toán song song.
 - (b) Độ phức tạp $O(n^2)$.
 - (c) Độ phức tạp $O(n)$.
 - (d) Khả năng xử lý long sequence.
2. Trong state space model, dạng convolutional có tính chất nào sau đây?
 - (a) Độ phức tạp $O(n^2)$.
 - (b) Không thể tính toán song song.
 - (c) Khả năng tính toán song song.
 - (d) Khả năng attention.
3. Trong structured state space model (S4), ma trận HiPPO được sử dụng để khởi tạo ma trận A vì:
 - (a) Khả năng tính toán song song.
 - (b) Tăng tham số để model học.
 - (c) Giảm tham số để model học.
 - (d) Tăng khả năng ghi nhớ sequence.
4. Trong state space model, biểu thức Dx_t trong công thức $y_t = Ch_t + Dx_t$ đóng vai trò gì?
 - (a) Activation function.
 - (b) LayerNorm.
 - (c) Skip-connection.
 - (d) BatchNorm.
5. Trong state space model, dạng recurrent chỉ có thể inference là nhận định:
 - (a) True
 - (b) False
6. Trong state space model, dạng convolutional phù hợp để train vì độ phức tạp $O(n)$ so với $O(n^2)$ của dạng recurrent là nhận định:
 - (a) True
 - (b) False
7. Trong state space model, dạng convolutional phù hợp để train vì khả năng tính toán song song là nhận định:
 - (a) True
 - (b) False
8. Trong state space model, dạng recurrent phù hợp để inference vì có khả năng tính toán song song là nhận định:
 - (a) True
 - (b) False

9. Trong state space model, dạng recurrent phù hợp để inference vì độ phức tạp $O(1)$ khi tạo ra từng token là nhận định:
- (a) True
 - (b) False
10. Đây là contribution của Mamba?
- (a) Khả năng tính toán song song ở dạng convolutional.
 - (b) Khả năng tính toán song song ở dạng recurrent.
 - (c) Khả năng tính toán song song ở 2 dạng convolutional và recurrent.
 - (d) Khả năng tính toán song song khi inference.
11. Đây là contribution của Mamba?
- (a) Khả năng tạo ra trọng số phụ thuộc vào input
 - (b) Khả năng tạo ra trọng số không dựa vào input.
 - (c) Khả năng tạo ra trọng số phụ thuộc vào label.
 - (d) Khả năng tạo ra trọng số không phụ thuộc vào label.

- *Hết* -