**Part I. Cơ sở cho Transfer Learning.**

**Mục lục:**

1. [Giới thiệu](https://dlapplications.github.io/2018-07-15-Transfer-Learning-Basic/#intro)
2. [Các định nghĩa](https://dlapplications.github.io/2018-07-15-Transfer-Learning-Basic/#first)
   1. Model
   2. Source Tasks và Target Tasks
   3. Transfer Learning
3. [Cơ sở](https://dlapplications.github.io/2018-07-15-Transfer-Learning-Basic/#second)
   1. Featuriser
   2. Fine-tuning
4. [Lợi ích và hạn chế](https://dlapplications.github.io/2018-07-15-Transfer-Learning-Basic/#third)
   1. Lợi ích
   2. Hạn chế?
5. [Kết bài](https://dlapplications.github.io/2018-07-15-Transfer-Learning-Basic/#conclusion)

**Mở đầu**

Bạn quá lười khi phải train lại network từ đầu?

Bạn quá mệt mỏi với việc tạo ra một end-to-end network mới để thực hiện một task gồm có nhiều phần khác nhau?

Bạn không có một bộ GPU mạnh và không muốn đăng ký AWS hay dùng Google Cloud? Hay bạn không có một bộ dataset lớn trong task mà mình phải thực hiện?

Well, đã đến lúc bạn hướng tới sự giúp đỡ của Transfer Learning, một kỹ thuật đưa đến những network đủ tốt chỉ với lượng dataset nhỏ trên cơ sở những network có sẵn.

Để tận dụng được các pretrained network này là một nghệ thuật được nghiên cứu từ thập niên 90. Khi Lorien Pratt [thực nghiệm](http://papers.nips.cc/paper/641-discriminability-based-transfer-between-neural-networks.pdf) lần đầu năm 1993 và sau đó viết lại nó dưới dạng một lý thuyết toán học (formal analysis) năm 1998.

Đây sẽ là bài thứ nhất của Transfer Learning, giúp cho các bạn có một cái nhìn tổng quan về các mặt của Transfer Learning trước khi đi vào thực hành ở bài viết kế tiếp ;)

**Prerequisite - Trước khi học về Transfer Learning, mình khuyến khích các bạn hiểu về:**

* Neural Network cơ bản.
* Dataset và các bước tạo ra một model.

**1. Giới thiệu**

Ngày xửa ngày xưa, khi nền văn minh của loài người chưa phát triển, các nhóm người nhỏ sinh sống trong những hang hốc. Khi con người biết trồng trọt, họ chuyển ra những đồng bằng sinh sống và tại đó, họ gặp những bộ tộc khác. Việc hiểu được nhau trở nên khó khăn khi số người càng ngày càng tăng.

Và thế là chúng ta phát minh ra ngôn ngữ, một cách để *truyền đạt* ý nghĩ của mình cho người xung quanh.

Việc nghiên cứu khoa học, đưa ra những ý tưởng mới thì quan trọng nhất là ***không làm lại những gì đã được làm rồi mà không làm tốt hơn được*** vì thời gian sẽ không cho phép sự lãng phí như vậy xảy ra. Đặc biệt là trong Deep Learning, một ngành phát triển nhanh đến chóng mặt hiện nay, những ý tưởng mình nghĩ ra chắc gì đã chưa có ai làm? Deep Learing lan tỏa đến mọi lĩnh vực, vì thế cái quan trọng là sử dụng những prior works sẵn có để tạo nên một model mới tốt hơn, vì chính việc này đã rất khó khăn và tốn thời gian rồi chứ không nói đến nghiên cứu lại từ đầu mọi thứ.

**2. Các định nghĩa**

**1. Model**

Chắc hẳn, nhiều bạn cũng đã biết về các model nổi tiếng, được train trên các dataset lớn (MNIST, CIFAR-100, ImageNet, …) và source code cũng như **Weights** của model được public cho cộng đồng (chủ yếu là trên GitHub). Chúng ta gọi những Model đi kèm Weights như vậy là một **Pretrained Model**.

Model mới sử dụng một phần hay toàn bộ pretrained model như một phần của nó để học một tasks mới được gọi là **Transfered Model**.

**2. Source Tasks và Target Tasks**

Những Pretrained Model như vậy thường được train trên một hoặc một vài bộ datasets nhất định, tương thích và cho accuracy cao với một task hoặc nhiều tasks (multi-task deep learning) nào đó mà nó được train. Chúng ta gọi các tasks mà pretrained model đó được train để thực hiện là **source tasks**.

Nhiệm vụ của chúng ta là tạo ra một model mới để thực hiện một hoặc nhiều tasks nào đó. Những tasks cần được thực hiện của model này có thể trùng hoặc không trùng với tasks mà pretrained model được train (thường thì sẽ không trùng), chúng ta gọi tasks này là **target tasks**.

**3. Transfer Learning**

Transfer Learning cũng chính là cách để các model truyền đạt cho nhau khả năng mà mỗi model có thể làm được. Một model có thể học trên source tasks nào đó và rồi pretrained model này được sử dụng cho model khác để model mới đó học trên target tasks nhanh hơn.

Cụ thể, **Transfer Learning** trong Deep Learning là một kỹ thuật mà trong đó:

* Một pretrained model đã được train trên source tasks cụ thể nào đó, khi đó một phần hay toàn bộ pretrained model có thể được tái sử dụng phụ thuộc vào nhiệm vụ của mỗi layer trong model đó.
* Một model mới sử dụng một phần hay toàn bộ pretrained model để học một target tasks và tùy vào nhiệm vụ của mỗi layer mà model mới có thể thêm các layer khác dựa trên pretrained model sẵn có.

Việc sử dụng pretrained model là một bước tiến lớn để những người đi sau tiếp bước những thành quả của các bậc tiền bối, tận dụng những pretrained model sẵn có để tạo ra những model mới phục vụ cho các target tasks cụ thể hơn, mang tính ứng dụng thực tiễn hơn.

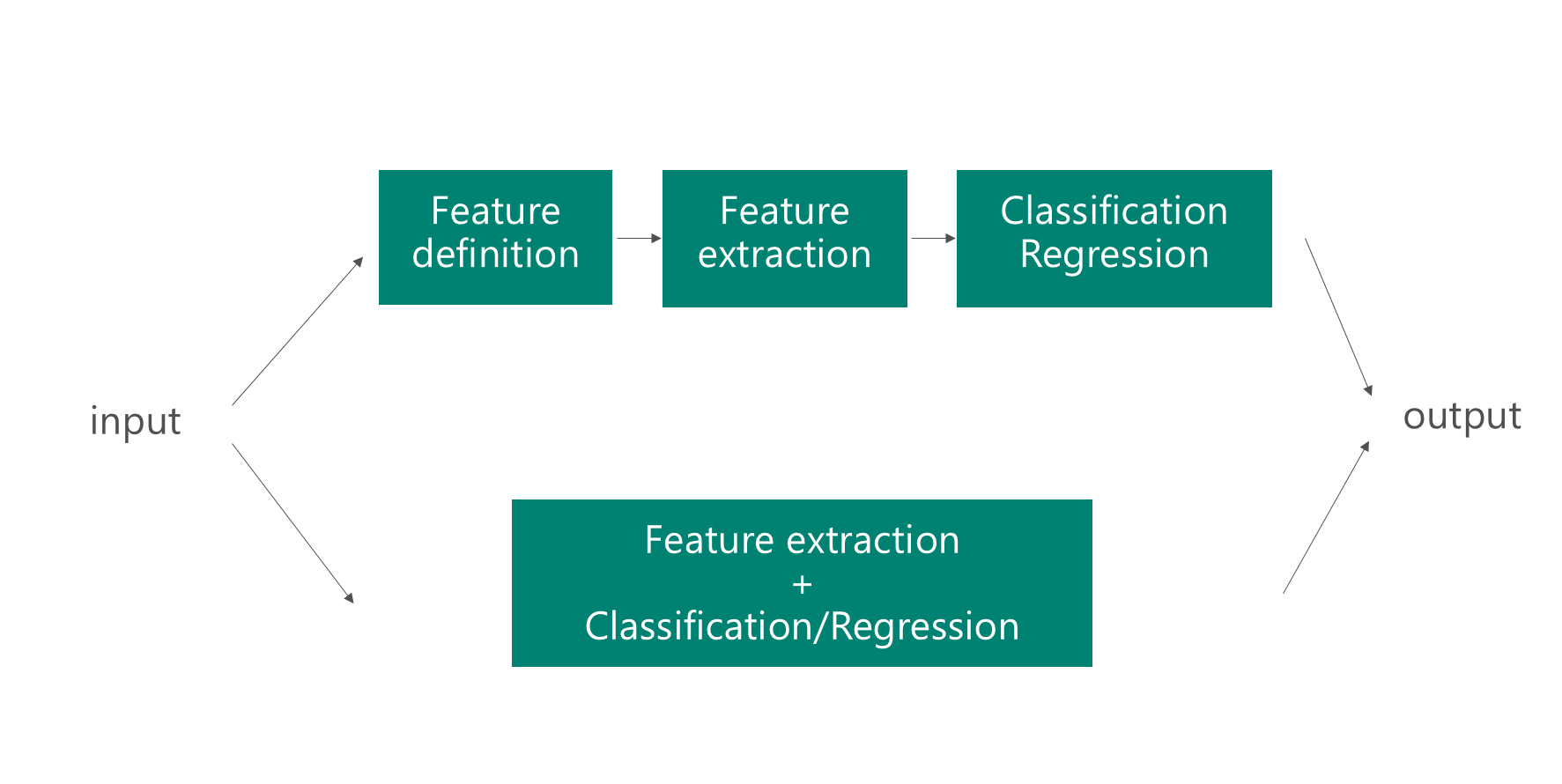
Đó không phải là sự sao chép ý tưởng, bản thân người tạo ra pretrained model đó public thành công của họ là vì hy vọng những người theo sau có thể tìm được những ích lợi từ các model đó, hay ít nhất là dùng nó để giải quyết các công việc của họ.

**3. Cơ sở**

**1. Featuriser**

Trước năm 2012, hầu hết mọi model AI đều được tạo thành bởi 2 stages độc lập với nhau:

* Feature Engineering: là quá trình dựa trên những hiểu biết của con người về vấn đề cần giải quyết (domain knowledge) để từ đó rút ra những đặc trưng (features) của dataset mà có thể giúp ích cho việc giải quyết vấn đề đó. Do đó các features này được gọi là hand-crafted features (nôm na là làm thủ công). Feature extractor là một phần của model dùng để trích xuất ra features nói chung.
* Classifier/Regressor: dùng các thuật toán Machine Learning để học và dự đoán các kết quả từ những features được tạo ra ở bước trên.

*So sánh phương pháp Featuriser của các hình thức cổ điển và Deep Learning hiện đại*

Các model Deep Learning tự nó đã kết hợp 2 stages này lại, các layer ở phần đầu của model được gọi là Feature Extractor và phần còn lại là để Classify/Regress các features từ phần Feature Extractor để tạo ra kết quả. Do đó phần Feature Extractor này có thể lấy ra được những features từ trong dataset một cách tự động trong qua trình học mà không cần con người định nghĩa các features như trong phương pháp Feature Engineering.

Có nhiều lý do khiến cho các Deep Networks hiện đại hiệu quả hơn các phương pháp cổ điển như trên (không kể đến các nguyên nhân chung như lượng dataset lớn hay khả năng tính toán song song, …), bao gồm:

* Deep Network là end-to-end trainable model: điều này cho phép các Feature Layers tự điều chỉnh những features mà nó cần trích xuất để phù hợp với tasks tương ứng (do kết quả backpropagation từ các Classifier/Regressor cho phép Extractor thích ứng theo yêu cầu của chúng), trong khi các cách cổ điển chỉ trích xuất features theo ý nghĩ của con người, một cách cố định.
* Khả năng mã hóa: Các bạn nào học Autoencoder cũng đã biết, các Layers đầu sẽ mã hóa dataset thành một tập latent variables và điều này có được là do Feature Layers đã lọc ra những features cần thiết và mã hóa nó (nói ngắn gọn là nén nó) thành một tập dữ liệu nhỏ hơn mà chỉ Decoder Layers của cùng Network mới giải được. Tương tự cho các Feature Extractor khác, nó có nhiệm vụ mã hóa input thành một mẫu phù hợp cho các layers tiếp theo, khi cần thiết nó sẽ thay đổi để đảm bảo những layers kế sẽ nhận input tốt nhất.
* Đặc trưng: Mỗi loại Deep Networks riêng đều có một nền tảng lý thuyết cụ thể lý giải tại sao loại Networks đó là Feature Extractor tốt trên loại dataset nào đó. Chi tiết cụ thể của điều này xin nhường lại những bài riêng.

Đây là nền tảng của Transfer Learning: chúng ta có thể sử dụng Feature Extractor đã được train để trích xuất các features cho model của chúng ta thay vì phải tạo ra một Feature Extractor mới và train lại từ đầu. Có thể nói là thay vì chạy bộ từ đầu đến cuối đường, chúng ta bắt taxi đến đoạn mà taxi không thể đi được thì chúng ta tự đi tiếp. Hãy hình dung sẽ ra sao nếu đoạn đường taxi đi được là gần hết quãng đường mà chúng ta cần đi? :)

**2. Fine-tuning**

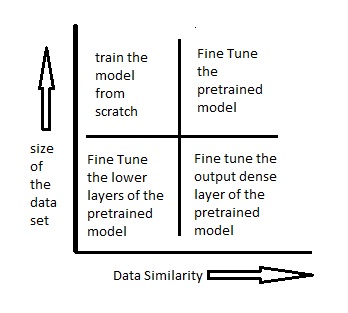
Để sử dụng pretrained model một cách hiệu quả, chúng ta cần phải có 2 điều sau:

* Thêm các layer phù hợp với target tasks của chúng ta, loại bỏ các layer của pretrained model mà chúng ta không dùng đến (việc này chắc các bạn cũng đã biết rồi), những phải khiến cho model **trở nên hiệu quả hơn**, đây là một vấn đề khó (rất rất khó) cần phải có những nghiên cứu chuyên sâu về từng layer và mục đích của chúng.
* Có chiến lược train thật tốt, điều này cũng không phải là dễ, vì nếu các bạn train không tốt thì sẽ làm mất đi tính hiệu quả của pretrained model và do đó giảm khả năng của model mà chúng ta đang train, thậm chí còn tệ hơn là train hết lại từ đầu.

Do đó, **fine-tuning** ra đời để giúp cho các bạn có chiến lược train hiệu quả trên transfered model của mình (điều đầu tiên theo như mình biết thì chưa tổng quát hóa được để tạo ra một kỹ thuật).

Fine-tuning không phải chỉ giúp cho các bạn điều chỉnh weights của transfered model cho phù hợp với target tasks. Nó không phải chỉ là **tinh chỉnh** như dịch nghĩa của fine-tuning mà xa hơn đó, nó đưa ra cách tối ưu để train cả phần pretrained model và phần mới trong transfered model nhằm đạt được accuracy cao trên target tasks, khiến cho 2 phần fit với nhau hoàn chỉnh thành một model mới.

Tóm lại, **fine-tuning** là việc train một transfered model nhằm mục đích tối ưu hóa accuracy của model này trên target tasks. Dưới đây là các chiến lược thường dùng:

*Phân loại chiến lược Fine-tuning*

* Khi dataset cho target tasks **lớn và tương tự** với dataset cho source tasks: đây là trường hợp lý tưởng, khi bạn có thể dùng weights của pretrained model để khởi tạo cho phần pretrained, sau đó train cả transfered model hay chỉ với phần được thêm vào, tùy bạn.
* Khi dataset cho target tasks **nhỏ và tương tự** với dataset cho source tasks: vì dataset là nhỏ, nếu train lại phần pretrained sẽ dẫn đến overfitting, do đó chúng ta chỉ train những layer được thêm vào với weights khởi tạo cho pretrained như trên.
* Khi dataset cho target tasks **lớn và khác biệt** với dataset cho source tasks: bởi vì dataset của chúng ta có sự khác biệt nên khi dùng weights từ pretrained model sẽ làm giảm accuracy vì sự khác biệt trong tasks và dataset, nhưng cũng chính vì dataset lớn nên việc train toàn bộ transfered model từ đầu là hiệu quả nhất, giúp cho model thích nghi tốt hơn với dataset này.
* Khi dataset cho target tasks **nhỏ và khác biệt** với dataset cho source tasks: đây là trường hợp khó khăn nhất, điều mà bạn nên làm có thể là:
  + Can thiệp vào pretrained model, thay thế những pretrained layer xa input để thích nghi với dataset mới (những high-level features sẽ thay đổi vào các low-level features đã được lấy từ các layer trước đó) nhưng **không được train các layer gần input của pretrained** vì dataset nhỏ sẽ không thể train được các layer này hiệu quả và các layer này chỉ trích xuất các features tổng quát từ dataset, sẽ không ảnh hưởng đến target task.
  + Tham khảo ý kiến chuyên gia hay tiền bối để biết thêm phương pháp.

**4. Lợi ích và hạn chế**

**1. Lợi ích**

* a. Thời gian

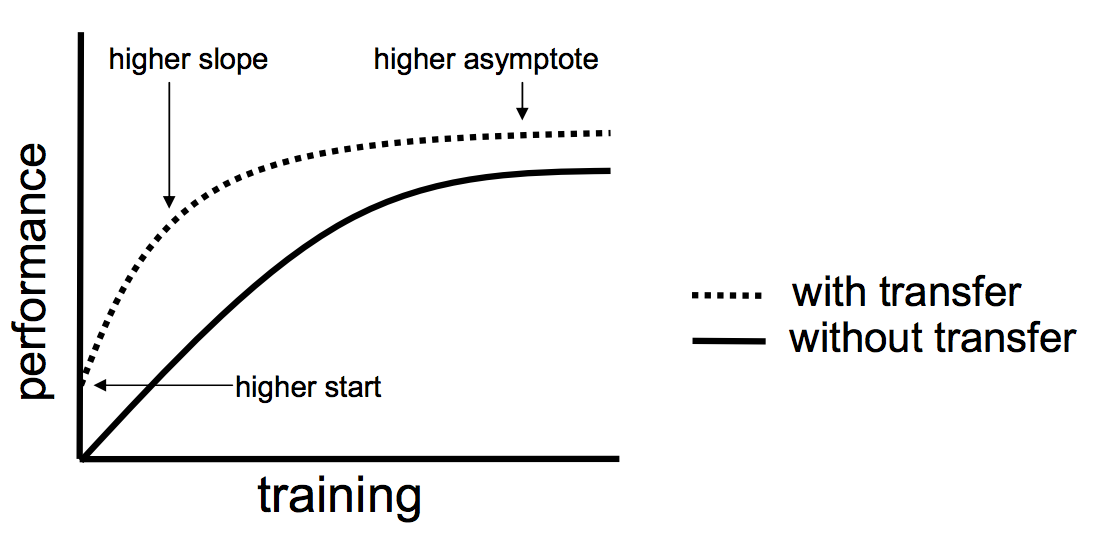
Việc sử dụng Pretrained Model bản thân nó không chỉ giúp giảm thời gian vào việc tạo ra một model mới để thực hiện một target tasks mà dựa trên một source tasks sẵn có, mà còn giảm thời gian train một model từ đầu vì Weights của phần source tasks đã có sẵn.

* b. Hiệu quả

Bạn nghĩ mình có thể tạo ra một model mới tốt hơn pretrained model trên cùng source tasks không?

Bạn có muốn tạo một model mới cùng source tasks mà phải có accuracy cao hơn các pretrained state-of-the-art không?

Nếu bạn quá lười để làm điều đó, pretrained model đã cung cấp cho bạn một accuracy cao ngay từ đầu, do đó khi train trên target tasks thì transfered model của bạn sẽ tiếp tục tăng accuracy này thay vì phải bắt đầu từ điểm có accuracy thấp hơn.

*So sánh tương quan hiệu quả của model train từ đầu và transfered model*

**2. Hạn chế?**

Transfer Learning không phải một kỹ thuật dễ sử dụng, nếu bạn sai sót trong quá trình transfer architecture của pretrained hay thêm/bớt không đúng layer thì khi train, accuracy sẽ **thấp không tưởng tượng được**, khi đó bạn sẽ phải kiểm tra lại quá trình sửa các layer hoặc làm lại từ đấu. ***Lưu ý***: khi bạn đạt accuracy thấp như vậy, chúng ta không gọi quá trình train là fine-tuning vì nó không phù hợp với định nghĩa.

Bạn chỉ có thể dùng Transfer Learning khi mà có pretrained model liên quan trực tiếp đến target tasks của bạn thôi, không phải pretrained model nào cũng có thể dùng để transfer vào target tasks mà bạn mong muốn được. Ví dụ bạn không nên dùng pretrained model cho hình ảnh màu để fine-tuning cho việc nhận diện chữ viết tay.

Trước khi dùng Transfer Learning, cũng như bao nhiêu hướng đi khác khi học tập cũng như nghiên cứu, bạn phải xác định rằng:

* Liệu có cần thiết phải transfer learning không?
* Chọn pretrained model nào là tốt nhất?
* Chọn dataset như vậy có phù hợp với pretrained model không?
* Chọn cách fine-tune nào là hiệu quả nhất?
* Liệu sau khi train thì model có accuracy cao hơn bình thường không?
* Vâng Vâng và Mây Mây …

Như các bạn có thấy, rất nhiều thứ để suy nghĩ trước khi transfer learning vì nhiều lý do khác nhau (Không phải source code nào cũng chạy được hay cho ra accuracy như trong paper? Source code thì public, nhưng bạn phải train để có pretrained model, vậy mục đích của transfer learning đang ở nơi nào? Reposistory cung cấp đầy đủ source code, pretrained nhưng bạn phải install ngôn ngữ mới để chạy và khi transfer learning thì bạn phải code trên ngôn ngữ đó, vì vậy phải học ngôn ngữ mới? … đủ thứ vấn đề bủa vậy bạn khi bạn định dùng pretrained model).

*Vậy nên là chúc các bạn tìm được pretrained model như ý nhé ;)*

**5. Kết bài**

Transfer Learning mang đến những model mới với độ chính xác cao trong thời gian ngắn, hầu hết các model dùng transfer learning được sử dụng trong các nghiên cứu về Computer Vision (CV), chú trọng vào việc trích xuất các features từ ảnh hoặc video một cách hiệu quả như một cách thay thế cho các phương pháp cũ (AKAZE, ORB, BRISK, …) và kết hợp những ý tưởng mới để tận dụng các features này (Object Detection, Object Recognition, Human Pose Estimation, …).

Transfer Learning cũng được sử dụng rất nhiều trong Natural Language Processing (NLP). Trên thực tế thì: nếu CV dùng Convolutional Network để trích xuất các features từ ảnh thì NLP dùng Word Embeddings như một cách để trích xuất các features từ các từ thành những vectors. Hiệu quả thực tiễn của Word Embeddings [cao hơn](https://www.datacamp.com/community/tutorials/lda2vec-topic-model) hẳn one-hot encodings về khả năng biểu diễn thông tin.