Vortraining

Autor und Herausgeber

Alexander Jung*, Department of Computer Science, Aalto University, Espoo, Finland (ORCID 0000-0001-7538-0990)
Editors: Konstantina Olioumtsevits (ORCID 0000-0003-2697-8659), Ekkehard Schnoor (ORCID 0000-0001-8252-1499), Aalto University, Espoo, Finland
*Corresponding author: alex.jung@aalto.fi

Zusammenfassung

Das Vortraining ist das Training eines Modells auf einem großen, generischen Datensatz, um Modellparameter zu erzeugen, die als Ausgangspunkt für eine spätere Lernaufgabe wiederverwendet werden. Die minimierte Verlust ist typischerweise selbstüberwacht, sodass keine manuelle Annotationsarbeit erforderlich ist. Bei einem großes Sprachmodell (LLM) minimiert das Vortraining die durchschnittliche Verlust der Vorhersage des nächsten Tokens über einen großen Textkorpus. Während das Transfer-Learning die Wiederverwendung dessen ist, was bei einer Lernaufgabe für eine andere gelernt wurde, ist das Vortraining die Phase, die das Wiederverwendete erzeugt, wobei zu diesem Zweck eine Stellvertreteraufgabe gewählt wird. Die Wiederverwendung nimmt zwei Formen an: die Feinabstimmung aller Modellparameter ausgehend von den vortrainierten oder das Einfrieren der vortrainierten Merkmalstransformation und das Trainieren ausschließlich der darauf aufgesetzten Schichten.

Definition

Bevor ein tiefes Netz darauf trainiert wird, Tumore in einigen hundert annotierten Scans zu erkennen, wird es auf Millionen gewöhnlicher Fotografien trainiert, die nichts mit Tumoren zu tun haben (Esteva et al., 2017). Dieser erste Schritt ist das Vortraining: das Training eines Modells auf einem großen, generischen Datensatz, um Modellparameter $\widehat{\mathbf{w}}$ zu erzeugen, die als Ausgangspunkt für eine spätere Lernaufgabe wiederverwendet werden. Der generische Datensatz des Tumorbeispiels ist ImageNet, ein Korpus von mehr als drei Millionen gelabelten Bildern bei seiner Veröffentlichung (Deng et al., 2009).

Die während des Vortrainings minimierte Verlust ist typischerweise selbstüberwacht (siehe selbstüberwachtes Lernen): Das Label eines Datenpunkts wird aus den rohen Daten selbst konstruiert, sodass keine manuelle Annotationsarbeit erforderlich ist. Bei einem großes Sprachmodell (LLM) ist das Label eines Datenpunkts das nächste Token des Textes, aus dem er ausgeschnitten wurde, und das Vortraining minimiert die durchschnittliche Verlust der Vorhersage des nächsten Tokens über einen großen Textkorpus (Brown et al., 2020). ImageNet trägt manuelle Labels, sodass das Tumorbeispiel die Ausnahme von der selbstüberwachten Regel bildet. Die während des Vortrainings

gelernten Modellparameter $\widehat{\mathbf{w}}$ werden anschließend durch Feinabstimmung für jeden aufgabenspezifischen Datensatz separat angepasst (Abb. 1).

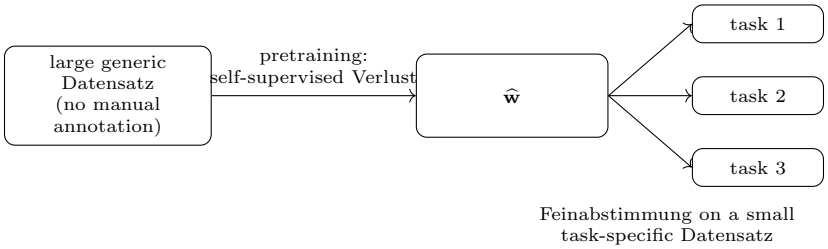


Abbildung 1: Das Vortraining, der beschriftete Pfeil, lernt Modellparameter $\widehat{\mathbf{w}}$ aus einem großen Datensatz, dessen Labels aus den rohen Daten konstruiert werden, sodass keine von Hand annotiert werden. Jede spätere Lernaufgabe unterzieht $\widehat{\mathbf{w}}$ auf ihrem eigenen kleinen Datensatz einer Feinabstimmung

Die Wiederverwendung nimmt neben der Feinabstimmung eine zweite Form an: Eine vortrainierte Merkmalstransformation $\phi\colon \mathcal{X} \to \mathbb{R}^d$ wird mit einem aufgabenspezifischen Modell verkettet. In einem tiefes Netz bilden die ersten Schichten die vortrainierte ϕ, und die Ausgabe-Schichten werden für jede Lernaufgabe separat trainiert (Abb. 2) (Yosinski et al., 2014). Dieselbe ϕ dient jeder Lernaufgabe, sodass generische Merkmale einmalig gelernt werden und jede spätere Lernaufgabe von einer informativen Initialisierung statt von Grund auf startet (Yosinski et al., 2014).

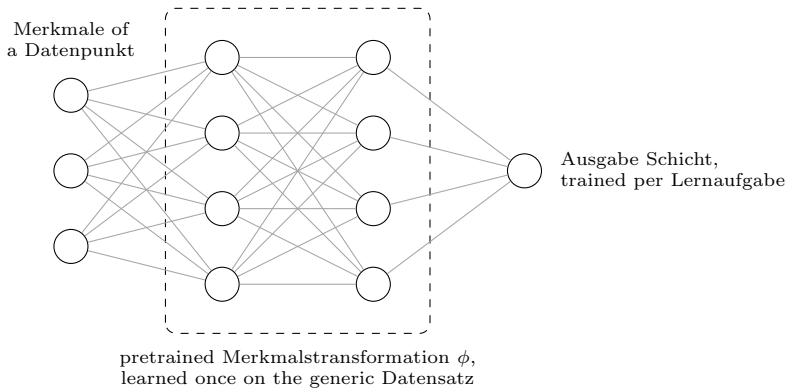


Abbildung 2: Ein tiefes Netz, dessen Eingabe links aus den Merkmale eines Datenpunkts besteht und dessen erste zwei Schichten die vortrainierte Merkmalstransformation ϕ (gestrichelter Kasten) bilden. Das Vortraining lernt diese Schichten einmalig auf dem großen generischen Datensatz. Die Ausgabe-Schicht wird auf dem kleinen Datensatz jeder Lernaufgabe separat trainiert, aufgesetzt auf dieselbe ϕ

Ein Ausschnitt einer Luftaufnahme der Wachau (Österreich), 2.4 km mal 1.2 km, wird in 4608 Kacheln von 25 m zerschnitten. Ein Parzellenregister versieht eine Kachel mit dem Label Weingarten, wenn mehr als die Hälfte ihrer Pixel auf Weingartenparzellen liegt, was bei einem Drittel der Kacheln der Fall ist. Die westliche Hälfte bildet den Trainingsdatensatz, die östliche Hälfte den Testdatensatz. Bei dieser Auflösung sind die Rebzeilen als sichtbare Streifen erkennbar, während die Farbe im Mittelwert einer Kachel wenig aussagt: Weingärten, Wiesen und Gärten sind allesamt grün (Abb. 3).

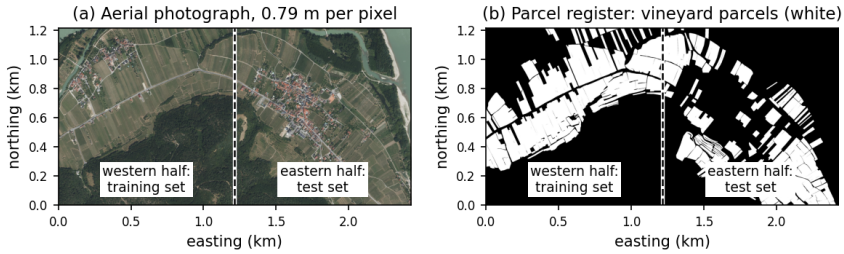


Abbildung 3: (a) Der Ausschnitt der Luftaufnahme der Wachau. (b) Das Parzellenregister, das die Kacheln mit Labels versieht: Eine Kachel gilt als Weingarten, wenn mehr als die Hälfte ihrer Pixel auf einer Weingartenparzelle liegt, hier weiß dargestellt. In beiden Teilbildern markiert die gestrichelte Linie die Aufteilung in den westlichen Trainingsdatensatz und den östlichen Testdatensatz. Fotografie: basemap.at (CC BY 4.0); Parzellen: INVEKOS 2025, AMA (CC BY 3.0 AT). Daten generiert durch `pythondemos/pretraining.py`

Das Vortraining nutzt allein die Fotografie, niemals die Labels. Der zentrale Block von 8×8 Pixeln einer westlichen Kachel wird ausgeblendet. Auf ein Convolutional Neural Network (CNN) aus zwei Faltungs-Schichten, jede mit acht 3×3-Filter und einer ReLU (ReLU), folgt eine lineare Ausleseschicht am ausgeblendeten Block. Dieses Netz wird darauf trainiert, die Farbe der ausgeblendeten Pixel im Mittelwert aus ihrer Umgebung vorherzusagen. Auf den östlichen Kacheln sagt es die ausgeblendete Farbe mit einem quadratischen Fehler von 0.00076 voraus, wohingegen die Antwort mit der Farbe im Mittelwert des restlichen Teils der Kachel 0.00296 ergibt, also das Vierfache (Abb. 4).



Abbildung 4: Acht östliche Kacheln von 25 m in drei Zeilen: die Kachel mit ausgeblendetem Zentrum (obere Zeile), die vom vortrainierten CNN aus der Umgebung vorhergesagte Farbe des Zentrums (mittlere Zeile) und die ursprüngliche Kachel (untere Zeile). Fotografie: basemap.at (CC BY 4.0). Daten generiert durch `pythondemos/pretraining.py`

Dies ist eine Vorwandaufgabe (Pretext Task): eine Lernaufgabe, deren Antwort an sich ohne Belang ist und deren Labels kostenlos mit den rohen Daten

mitgeliefert werden, einzig zu dem Zweck gestellt, dass ihre Lösung die Schichten dazu zwingt, etwas zu repräsentieren. Eine zweite Vorwandaufgabe trainiert dasselbe Netz darauf zu erkennen, ob eine Kachel um 0, 90, 180 oder 270 Grad gedreht wurde. Die Richtung der Rebzeilen und der Schatten beantwortet diese Frage. Das Netz erkennt die Drehung bei 37 Prozent der östlichen Kacheln, gegenüber 25 durch bloßes Raten.

Die beiden Faltungs-Schichten werden anschließend eingefroren: Sie bilden die Merkmalstransformation ϕ, und ihr Ausgabe auf der ursprünglichen Kachel, gemittelt über ein 4×4-Gitter, liefert 128 Merkmale. Die logistische Regression auf diesen Merkmale ist ein linearer Klassifikator für Weingarten gegenüber Nicht-Weingarten. Er wird auf nur 8 und höchstens 128 gelabelte Datenpunkte der westlichen Hälfte trainiert, und seine Genauigkeit wird auf der östlichen Hälfte gegenüber der Mehrheits-Baseline von 0.69 gemessen, der Genauigkeit der ständigen Antwort „Nicht-Weingarten“ (Abb. 5). Die östliche Hälfte wird während des gesamten Trainings zurückgehalten, sodass die auf ihr gemessene Genauigkeit schätzt, wie der linearer Klassifikator auf Kacheln abschneidet, aus denen er nicht gelernt wurde. Die westliche Hälfte enthält weit mehr als 128 gelabelte Kacheln; es werden nur wenige verwendet, da dies der Bereich ist, in dem das Vortraining über das Ergebnis entscheidet.

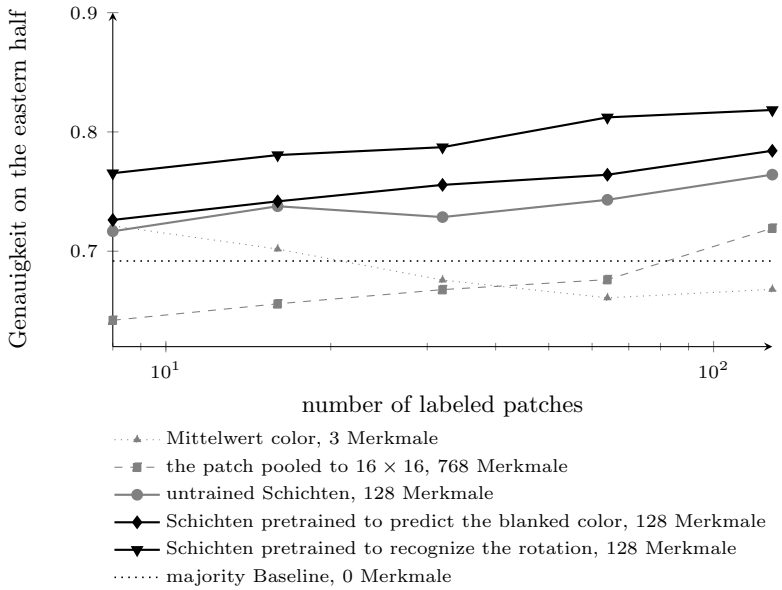


Abbildung 5: Genauigkeit eines linearer Klassifikators von Weingartenkacheln auf der östlichen Hälfte in Abhängigkeit von der Anzahl gelabelter westlicher Kacheln für fünf Merkmal-Mengen: die Farbe im Mittelwert (gepunktete graue Kurve, gefüllte Dreiecke), die gepoolte Kachel (gestrichelte graue Kurve, gefüllte Quadrate), untrainierte Schichten (durchgezogene graue Kurve, gefüllte Kreise), auf die Vorhersage der ausgeblendeten Farbe vortrainierte Schichten (durchgezogene schwarze Kurve, gefüllte Rauten) und auf das Erkennen der Drehung vortrainierte Schichten (durchgezogene schwarze Kurve, umgekehrte gefüllte Dreiecke). Die gepunktete schwarze Linie ist die Mehrheits-Baseline. Parzellen: INVEKOS 2025, AMA (CC BY 3.0 AT). Daten generiert durch `pythondemos/pretraining.py`

Bei so wenigen Labels neigt ein linearer Klassifikator auf den Pixeln selbst zu Überanpassung: Seine Vapnik-Chervonenkis-Dimension (VC-Dimension) beträgt die Anzahl der Merkmale plus eins (Shalev-Shwartz and Ben-David, 2014, Thm. 9.3), und die Anzahl der Labels, die die Generalisierung erfordert, wächst proportional dazu (Shalev-Shwartz and Ben-David, 2014, Thm. 6.8). Er verfügt über 768 Merkmale – die auf 16×16 Pixel in drei Farben gepoolte Kachel – und höchstens 128 Kacheln, um sie daraus zu lernen, und seine Genauigkeit bleibt unter dieser Baseline, bis 128 Labels verfügbar sind, womit er 0.72 erreicht. Auch die Farbe im Mittelwert einer Kachel verbleibt bei dieser Baseline, mit 0.67 bei 128 Labels. Die Kontrolle, die jede Aussage zum Vortraining benötigt, besteht aus denselben zwei Schichten mit ihren zufälligen anfänglichen Gewichte, niemals trainiert: Sie erreichen 0.76 bei 128 Labels, da zufällige Faltungs-Filter mit Pooling bereits auf Textur ansprechen (Saxe et al., 2011). Beide vortrainierten Schichten liegen bei jeder Anzahl von Labels über dieser Kontrolle, jedoch mit

sehr unterschiedlichem Abstand: die auf Farbe vortrainierten um weniger als 0.01 unterhalb von 32 Labels und um etwa 0.02 ab 32, womit sie 0.78 bei 128 erreichen, die auf Drehung vortrainierten durchweg um 0.04 bis 0.07, womit sie 0.82 erreichen.

Daraus folgen zwei Schlüsse. Erstens hilft das Vortraining, wenn der gelabelte Datensatz zu klein ist, um das zu lernen, was die Vorwandaufgabe den Schichten bereits beigebracht hat: Der Vorsprung gegenüber der Kontrolle ist bei den kleinsten Anzahlen von Labels am größten und verringert sich, je mehr hinzukommen. Zweitens entscheidet die Wahl der Vorwandaufgabe darüber, wie viel gewonnen wird. Das Erkennen der Drehung ist nicht möglich, ohne die Richtung der Streifen zu repräsentieren – genau das, was einen Weingarten von einer Wiese unterscheidet –, sodass es die Merkmale liefert, die der linearer Klassifikator benötigt. Die Vorhersage der ausgeblendeten Farbe lässt sich allein anhand der lokalen Farbe bewerkstelligen, was sich angesichts der Genauigkeit der Farbe im Mittelwert hier als wenig nützlich erweist; diese Aufgabe erzwingt also weniger von dem, was der linearer Klassifikator benötigt, selbst wenn das Netz sie viermal besser löst als die Mittelwertbildung. Dieselbe Überlegung gilt für ein LLM: Die Vorhersage des nächsten Tokens ist nicht möglich, ohne zu repräsentieren, worum es in dem Text geht. Die Zahlen werden durch `pythondemos/pretraining.py` berechnet. Synonyme: Pretraining, Pre-Training.

Siehe auch: Transfer-Learning, Feinabstimmung, selbstüberwachtes Lernen, Grundlagenmodell, Multitask-Lernen, tiefes Netz.

Querverweise

Literatur

1. Brown et al. (2020). Language Models are Few-Shot Learners. In: Adv. Neural Inf. Process. Syst., pp. 1877–1901.

2. Esteva et al. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), pp. 115–118.

3. Deng et al. (2009). ImageNet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255.

4. Saxe et al. (2011). On Random Weights and Unsupervised Feature Learning. In: Proceedings of the 28th International Conference on Machine Learning (ICML), pp. 1089–1096.

5. Shalev-Shwartz and Ben-David (2014). Understanding Machine Learning: From Theory to Algorithms. Cambridge Univ. Press, New York, NY, USA.

6. Yosinski et al. (2014). How Transferable Are Features in Deep Neural Networks?. In: Adv. Neural Inf. Process. Syst. (NIPS), pp. 3320–3328.