technische universität dortmund

fakultät für informatik

Masterarbeit – Zeit-Effizientes Training von Convolutional Neural Networks

Zwischenstand

Jessica Bühler

6. November 2019

Zusammenhang Hardware und Zeitersparnis

In der Praxis wird für das Trainieren eines CNNs in der Regel eine Grafikkarte genutzt. Für die Bearbeitung meiner Masterarbeit stehen mir momentan zur Verfügung:

- Geforce 750Ti mit 2 Gb mit CUDA-Version 5.0
- Geforce 1070 mit 11 Gb mit CUDA-Version 6.1
- Lehrstuhlserver mit ?? mit ?? Gb mit CUDA-Version ??

Die großen Unterschiede im Speicher machen es notwendig unterschiedlich große Netze zu trainieren. Wie gross ist der Unterschied im Gewinn bei der Trainingszeit?

Übersicht I

Code

https://bitbucket.org/lph_tools/prunetrain/src/master/

Was ist Pruning?

- Modell Pruning reduziert die Zahl der zu lernenden Parameter in einem dichten Netzwerk, sodass die Inferenzkosten und die Speicherlast reduziert werden.Dabei soll so wenig wie möglich Accuracy verloren gehen.
- Hauptziel ist als das Verbessern der Inferenz, aber auch das Training kann signifikant schneller werden.

Pruning währenddem Training – bisherige Vorgehensweise

- Man kann durch das Hinzufügen von Normalisationstermen prunen. Hierfür kann eine L_1-Norm oder ein sog. “group lasso” verwendet werden.
- gropu lasso verwendet L_1 oder L_2 Normen von Gruppen von Gewichten für strukturelles Pruning. Daher werden vom Optimierungsprozess hier kleine absolute ?? Werte für Gewichte oder Gruppen von Gewichten bevorzugt. Dies für zu einem nicht mehr dichten Netz.
- Durch das beibehalten der nicht mehr dichten Sturuktur kommtes allerdings nicht zu Gewinnen im Trainingsprozess.
- wird die dichte Struktur nicht beibehalten so wurde bisher komplexes Dataindexing notwendig, welches die Performance verkleinert.

Pruning währenddem Training – neue Vorgehensweise

- Eine Vorarbeit rekonfiguiert das CNN genau einmal währenddem Training und arbeitet danach mit dem kleineren Modell. Allerdings ist der Rekonfigurierungszeitpunkt vorher nicht festgelegt, was diese Herangehensweise problematisch oder sogar kontraproduktiv macht.
- Daher nun Prune Train mit:
  - Beschleunigung des Trainingsmechanismus des CNN welches das Model währenddem Training pruned.
  - Der Prune-Prozess startet bereits bei der ersten Epoche
  - Es wird ein sog. “group lasso” Regulierungsverfahren als Basis für die Linienausdünnung benutzt. Die Gewichte werden regelmässig gepruned.
  - Das CNN wird regelmässig reconfiguriert um das Modell kleiner und so das Training schneller zu machen.

Die 3 zentralen Optimierungsverfahren von PruneTrain

- eine systematische Methode zur Berechnung des group lasso Regularisierung Sanktions Koeffizienten beim Beginn des Trainings.
- Kanal union, ein Speicheraufruf kosteneffizientes und Index-freies Kanal Pruning Verfahren für moderne CNNs mit Kurzschlussverbindungen.
- Ein dynamische Mini-Batch Adjustment, dass die Größe des Mini-Batch anpasst. Dies geschieht durch beobachten des Speicherkapazitätgebrauchs einer Trainingsiteration nach jeder Pruning reconfiguration.

group lasso Regularisierung Sanktions Koeffizienten

- Der group lasso Regularisierung Sanktions Koeffizienten ist ein Hyperparameter, der einen Trade-off zwischen der Modellgröße und der Accuracy bildet.
- Voherige Arbeiten suchen nach einem geeignetem Sanktionsmaß, was das Einbeziehen des Prunings vom Anfang des Trainings sehr teuer macht.
- Unser Mechanismus kontrolliert die Group lasso Regularisierungstärke und erreicht eine hohe Modellpruningrate mit nur einem kleinen Einfluss auf die Accuracy bei nur einem Trainingsdurchlauf.

Kurzschlussverbindungen und Union Channel

- Kurzschlussverbindungen werden in modernen CNNs häufig genutzt.
- Prunning aller genullten Kanäle solcher CNNs brauchen regelmässige Tensor Umordnung um die Kanalindizes zwischen den Schichten zu matchen. Dies vermindert die Performance.
- Diese Umordnung wird durch den Channel Union Algorithmus vermieden. Daher folgt eine 1.9 fache Beschleunigung des Convolutional Layetrs.

Dynmaisches Minibatch Adjustment

- Dynmaisches Mini Batch Adjustment kompensiert die verminderte Datenparallelität aufgrund des kleineren geprunten Modells durch Erhöhung der Mini-Batch Größe.
- Dies sorgt sowohl für bessere Ausnutzung der Hardware ressourcen als auch zur Reduzierung des KOmmunikation overheads durch eine Verminderte Modell Update Frequenz.
- Beim Erhöhen der Mini-Batch Größe wird auch die Lernrate mit demselben Verhältnis erhöht, um die Accuracy nicht zu verändern.

Pruning Algorithmen

Pruning Algorithmen sind

- strukturiert oder
- unstrukturiert.

Unstrukturiertes Pruning kann die maximale Modellverkleinerung erarbeiten braucht aber feine Indexing. Daher ineffizient.
Struktuiertes Pruning entfernt oder reduziert feines Indexing und sorgt für ein besseres Ausnutzen der Hardware und realisiert daher einen Performancegewinn.

Strukturiertes Pruning I

In diesem Paper werden zwei Arten von strukturiertem Pruning gezeigt:

- Versuch und Irrtum basiertes strukturelles Pruning: Startes mit einem vortrainiertem dichten Modell und versuche dann Gewichte in einem strukturellen Vorgehen zu entfernen, sodass eher ganz Kanäle entfernt werden als einzelne Gewichte. Nicht wichtige Kanäle werden basierend auf ihrem Wert der Gewichte oder wegen Hinweise durch die Regression entfernt.Die entfernten Knäle können wieder eingebunden werden, falls der Verlust an accuracy zugroß ist. Diese Verfahren ist zwar effective, allerdings vergrössert sich der Suchraum mit der Komplexität des Modells. Dadurch kann die Pruning Zeit signifikant anwachsen. Ausserdem sorgt diese Verhfaren durch die Vewendung eines vortrainierten Netzes zu keiner Zeiteinsparung beim Training.

Strukturiertes Pruning II

- Ein alternativer Mechanismus nutzt Parameterregularisierung. Dies optimiert den Trainingsfehler und sorgt simultan dafür, dass die absluten Werte der Gewichte oder von Gruppen von Werten gegen Null gehen. Group lasso Regularisierung wird benutzt um die Gewichte strukturell zu verdünnen durch die Zuweisung einer Regularisierungssanktion zu L_2 Normen von Gruppen von Gewichten Der Regularisierungsbasierte Pruning Ansatz fügt noch ein Regularisierungsfunktion zum Fehlermass hinzu und minimiert bestimmte Gewichte oder Gruppen davon durch die Backpropagation. Eventuell können Gewichte sogar ganz auf Null gesetzt und dann gepruned werden.

Mathematische Grundlagen – Baseline Pruning

$$\underset{W}{min}\left(\frac{1}{N}\sum_{i=1}^{N} l(y_i, f(x_i, W)) + \sum_{g=1}^{G} \lambda_g \cdot ||W_g||_2\right)$$

- $\frac{1}{N}\sum_{i=1}^{N} l(y_i, f(x_i, W))$ Standard-Kreuzentropie
- $\sum_{g=1}^{G} \lambda_g \cdot ||W_g||_2$ group lasso Regulierungsterm
- $f(x_i)$ Vorhersage des Netzwerks auf Eingabe x_i
- W Gewichte
- l Verlustfunktion der Klassifikation und Grundwahrheit y_i
- N Minibatchgröße
- G Zahl von Gruppen
- λ Verdünnungskoeffizient

Mathematische Grundlagen – Group lasso

Design

$$\lambda \cdot \sum_{l=1}^{L} \left(\sum_{c_l=1}^{C_l} ||W_{c_l,:,:,:}||_2 + \sum_{k_l=1}^{K_l} ||W_{:,k_l,:,:}||_2 \right)$$

Design eines speziellen Groupo Lasso Regulierers, der Gewichte jedes Kanals (Input oder Output) und jeder Schicht gruppiert. λ wird als einziger globaler Regularisierungsfaktor gewählt, da so der Fokus auf dem Vermindern der Rechenzeit liegt und nicht auf der Modellgröße. Dies hat zur Folge, dass vorallem große Features verdünnt werden, was zu einer größeren Verminderung der Rechenleistung führt.

Setup des Regularisierungssanktionskoeffizienten

Um die Lasso Group Regularisierung vom Anfang des Trainings zu benutzen sollteder Koeffizient λ sinnvoll gewählt werden. Dies sorgt für eine hohe Vorhersageaccuracy und einer hohen Pruning Rate. Um zeitintensives Hyperparametertuning zu vermeiden wird hier eine neue Methode eingeführt:

$$LPR = \frac{\lambda \sum_g^G ||W_{g,:}||}{l(y_i, f(x_i, W)) + \lambda \sum_g^G ||W_{g,:}||)}$$

Berechnet wird dies durch setzen von zufälligen Werten, mit denen die Gewichte initialisiert werden. LPR wird einmal berechnet und dann bis zum Ende weiter benutzt.

Schichtentfernung durch Überlappende Regulierungsgruppen

Ist nicht notwendig, da Schichten die nicht relevant sind eh irgendwann gepruned werden.

Frühes Gewichtspruning

Gewicht auf 0 -> Gradient auf 0 -> Erholung sehr unwahrscheinlich -> Daher kann ein auf 0 gesetztes Gewicht gepruned werden.

Robustness des Reconfigurationsintervalls

Nach jedem solchen Intervall werden Input und Outputkanäle die 0 sind gepruned. Um ein Missverhältnis zwischen den Dimensionen zu verhindern wird nur die Verbindung von 2 verdünnten Kanälen von 2 aufeinanderfolgenden Schichten gepruned. Alle Trainingsvariablen bleiben gleich. Das Reconfigurationsintervall ist ist der einzige zusätzliche Hyperparameter. Zu gross gewählt würde der Intervallparameter zu wenig Zeitverbesserung bringen. Zu klein gewählt könnte er die Lernqualität beeinflussen.

Channel Union vs. Channel Gating

- Channel Union mildern das Vanishing Gradient Problem und öerlauben so tiefe Netze und hohe Accuracy.
- Hier werden 2 Ansätze benutzt, um die passende Dimesionierung zu gewährleisten:
  - Channel Gating Schichten
  - Channel Union
- Channel Gating braucht die schon erwähnt Umordnung der Tensoren, Channel Union nicht.
- Channel Union prunt nur die Verbindungen von verdünnten Kanälen von allen benarchbarten Schichten in einer Rest-Phase.
- Channel Union wird benutzt da es mehr Rechenzeit spart als Channel Gating

Dynamisches Mini-Batch Adjustment

Bei der Evaluierung zeigt sich, dass die erhofften Effekte des Mini Batch Adjustment passieren.

Evaluierung

We first present our evaluation results on CIFAR and ImageNet in Tab. 1. We report 4 metrics: the training and inference FLOPs (FP operations), measured training time, and validation accuracy. Training time does not include network architecture reconfigura- tion time, which we do optimize and occurs only once in many epochs. We compare the training results of ResNet and VGG using PruneTrain with the dense baseline. We use the same number of training iterations for both the dense baseline and PruneTrain to show the actual training time saved by PruneTrain. We use 182 epochs [15] and 90 epochs to train CNNs on CIFAR and ImageNet, respectively. For ResNet32 and ResNet50 on CIFAR10, PruneTrain reduces the training FLOPs by 50the dense baseline. The compressed models after training show only 34ResNet50, respectively. The results of ResNet32/50 on CIFAR100 show similar patterns, which exhibits the robustness of PruneTrain, given that CIFAR100 is a more difficult classification problem. For CIFAR100, PruneTrain reduces the training and inference FLOPs by 32losing only 1.4pared to the dense baseline. These results show that PruneTrain reduces

Conclusion