What should be said

Stanislav Arnaudov

<2020-05-31 Sun>

1 Einleitung

Hallo, ich bin Stanislav und heute präsentiere ich unseren finalen Vortrag zu Praxis der Forschung.

1.1 Grund Idee

Ganz am Anfang, wenn ich die grobe grundlegende Idee unseres Projekts wiedergeben soll: wir haben es versucht, Machine-Learning Methoden bei Computatinal Fluid Dynamics (oder CFD) anzuwenden. Ein bisschen konkreter, wir wollen untersuchen, in wie fern können Deep Neural Networks (DNNs) beim Durchführen und Visualisieren einer Strömung-Simulation angewandt werden.

1.1.1 Machine Learning

Machine Learning und ins besondere DNNs werden heutzutage in ganz verschiedenen Anwendungsfällen verwendet. In den letzten Jahren ist es sogar möglich, Bilder mit DNNs zu generieren und unsere Arbeit ist in genau diese Richtung orientiert. Also mit DNNs gewisse Information (im Bildform) abarbeiten und anhand davon ein neues Bild erzeugen.

1.1.2 CFD

Auf der anderen Seite haben wir CFD. Bereich wo anhand von mathematischen Modellen (in meinten Fällen Differential Gleichungen) verschiedenen Simulationen von Flüssigkeiten durchgeführt werden. Hier werden spezielle Solvers verwendet, die unter Umständen ziemlich viel Rechnerleistung benötigen. Angesichts davon schlagen wir mit unserer Forschung eine alternative Strategie für das Durchführen von Simulationen vor und zwar fokussieren

wir uns auf Fällen, wo die visuelle Repräsentation den Ergebnissen auf dem Vordergrund steht.

2 Forschungsprojekt

Um zu erklären, was genau das Forschungs-Projekt enthält, werden wir die folgenden drei Fragen beantworten - “was”, “warum” und “wie” es gemacht wurde. Die Antworten hängen einigermaSSen zusammen, aber verseuchen wir da ganze Schrittweise durchzugehen.

2.1 Was

Also erstmal “was?” - in ganz einfachen Wörtern, wir wollen die Strömung in einem Kanal um ein Objekt vorhersagen. Und zwar mit einem neuronalen Netz gegeben eine ungewöhnliche Repräsentation vom Raum. Ich sage mehr dazu später, jetzt stellen wir das Problem ein bisschen anschaulicher vor.

Wir haben die Folgenden Situation, ein Kanal und ein Objekt in der Mitte. Wir beschränken uns auf dem 2D Fall. An einer Seite hier gibt’s eine EinflieSS-Bedingung, also da kommt die Flüssigkeit rein und an der anderen Seite flieSSt die raus. Mit dieser Einstellung haben wir auch einige Parameter für die Simulation, nämlich EinflieSSgeschwindigkeit, Viskosität und Dichte vom Fluid. Diese sind also die Zahlen die das eine konkrete Instanz vom Problem so zu sagen bestimmen.

Das Ziel dann ist, die Simulation durchzuführen und die Geschwindigkeit und den Druck in jedem Punkt vom Raum zu bestimmen für eine gewisse Zahl von Zeitschritten. Also die Lösungen für jeden Zeitschritt zu bestimmen.

In unserem Anwendungsfall wollen wir nicht explizit die Roh-Daten für die Lösungen, sondern eine visuelle Repräsentation davon erzeugen. Also, spricht, Bilder, die durch verschiedene Pixel-Farben, verschiedene Werte vom Lösungsraum kodiert. Pro Zeitschritt haben wir drei Solche Bilder - zwei Geschwindigkeit-Felder (also in die zwei Richtungen) und ein Druck-Feld.

Wie schon erwähnt, wir wollen dies mit einem DNN vorhesagen. Gegeben also die Bilder von einem Zeitschritt, die Bilder für den nächsten zu erzeugen. Natürlich möglichst genau. Das DNN soll auch in der Lage sein, die Parameter der Simulation betrachten und die Lösungen entsprechend anpassen.

Mit unserer Forschung wollen wir also untersuchen, ob dies möglich ist und ob ein Netz die Parameter die Simulation generalisieren kann und Simu-

lationen mit ungesehen Parameter gut vorhersagen. Und falls ja - inwiefern (also die Ergebnisse Quantifizieren)

2.2 Warum

Jetzt kommen wir zu das "warum?" - Normalerweise wird eine solche Simulation wird mit einem numerischen Solver gelöst. Die Visualisierung vom Ergebnissen kommt dann als ein separater Schritt. Schematisch können wir das ganze so darstellen.

Am Anfang ist selbst das Problem, das gewisse Parameter besitzt. Dann kommt der Solver, der reelle Zahlen als Ergebnisse für jeden Zeitschritt ausrechnet und dann kommt die Visualisierung (was auch ein Tool), die die letztendliche Bilder erzeugt. Offensichtlich diese Pipeline stellt eine kompliziertes System, wo das Workflow nicht trivial ist, für jede Phase es verschiedene Parameters gibt und die notwendige Rechnerleistung nicht zu übersehen ist.

Zu bemerken ist, dass wir nur an den Bildergebnisse interessiert sind. Also, als Menschen, die das Simulation nur "sehen" wollen, machen für uns Zahlen kaum Sinn. Natürlich gibt's Fälle wo die Zahlen gebraucht sind das wichtigste Ergebnisse ausmachen, aber für unsere Forschung, haben wir uns fokussiert, die Bilder hier möglichst schnell und effektiv zu generieren. Unser Ziel ist also diese ganze Pipeline hier mit einem Netz zu Ersetzen.

- Damit versuchen wir alle Aufgeben beim Trainiren zu lernen und nicht explizit zu modellieren.
- Wir lassen uns sagen, dass die Ergebnisse nicht absolut exakt sein müssen als die Bilder für ein Mensch ziemlich gleich aussehen können ohne komplette Überlappung.
- Das ganze System ist auSSerdem performanter weil DNNs sehr schnell Bilder abarbeiten können und sich für solche Aufgaben etabliert haben.

2.3 Wie

Letztendlich können wir unser Ansatz genauer erklären (also die "wie?" Frage beantworten). Hier gehen wir kurz durch die Schritten, die erledigt werden müssten, damit wir unsere Ziele erreichen könnten.

Es wurde erwähnt, dass die Simulation drei Parameter besitzt. Wir haben aber uns entschieden nicht ein holistisches Netz zu entwickeln, das alle drei Parameter behandeln kann, sondern drei kleineren, die das Problem schrittweise eingehen. Diese sind

- konstantes Modell - es werden gar keine Parameter betrachtet. Baseline Modell gedacht als Proof-of-Concept. Hier ist die Generalisierung nur in die Zeit. Also, lerne auf einem Teil einer Simulation, vorhersage den Rest.
- Geschwindigkeit Modell - Netz, dass nur die EinflieSS-Geschwindigkeit behandeln kann.
- Viskosität-Dichte Modell - selbsterklärend, hier können die anderen zweit Parameter variiert werde.n

Wir waren auSSerdem daran interessiert, ob der Druckfeld Unterschied bei der Leistung der Netzen macht. Deswegen haben wir pro Modell Typ, zwei Modellen trainiert - ein das den Druckfeld benutzt (bei der Ein- und Ausgabe) und ein das nicht. Mit der Evaluierung sehen wir dann, ob zwei solche Modelle sich wesentlich Unterscheiden.

2.3.1 Daten Erzeugung

Data sets:

Als grundsätzlich unsere Forschung ein Maschine Learning Projekt ist, brauchen wir zunächst echte Daten, mit denen wir Modelle trainieren können. Diese haben wir mit der gezeigten Pipeline erzeugt. Das beschriebenen Problem (also die Strömung im Kanal) wird durch die so genannte Navier-Stokes Gleichungen beschreiben. Das heiSSt, es muss eine Partielle Differentialgleichung (PDE) gelöst werden um mit den Lösungen anzukommen. Dies wurde mit HiFlow3 gemacht, was ein Bibliothek zum Lösen von PDEs. Die Lösungen wurden dann als Bilder mit ParaView (ein Visualisierung-Tookkit) visualisiert.

Für die Visualisierung haben wir uns Graustufenbilder gewählt. Frühe Experimente haben gezeigt, dass das Trainieren mit Farbige Bilder (also drei Kanale pro Bild) wesentlich schwieriger ist.

Die benutzte Modell-Gleichung für die Daten-Erzeugung zeigt uns welches Problem das Netz selbst quasi kodieren muss um die Daten vorherzusagen. Wir sagen deswegen, dass unsere Netz sich spezifisch zu den Navier-Stokes Gleichungen für inkompressibele Fluid-Strömung ausrichtet.

Wir haben uns auch drei Sets von Simualtionsbilder erzeugt wo verschiedenen Parameter variiert wurden. Also zum Beispiel für das erste Modell gibt's nur eine Simulation mit festen Parametern und für das Zweite gibst eine Reihe von Simulationen mit verschiedenen EinflieSS-Geschwindigkeiten.

Parameter Wahl : Die Wahl von Parametern für die Simulationen ist nicht zufällig gewesen. Wir haben die Reynoldszahl der Strömung betrachtet und diese bestimmt die Art der Strömung und zeigt ob die Laminar, Turbulent oder etwas dazwischen ist. Die Simulationensparamter wurden so gewählt, dass die Reynoldszahl im Bereich von 90 bis 450 liegt. Diese Bedeutet, dass die Strömung nicht turbulent ist, aber gewisse interessante Wirbelstrukturen sind sichtbar (wie zum Beispiel die bekannte Karmansche WirbelstraSSe hier). Die Idee hinter dieser Entscheidung ist zu sehen ob die Netze mit nicht-trivialen Simulationen umgehen können.

Test train split: Die Datensätze wurden gesplittet in Training-Set und Test-Set. Wichtig hier zu sagen, dass in den Test-Sets gab's Simulationen mit Reynoldszählen, die für keine Simulationen in Training-Set zu finden ist. Also die Daten sind so gestaltet, so dass wir wirklich sehen können, ob das Netz die Parameter generalisieren und ungesehene Simulationen vorhersagen kann.

2.3.2 Netze Architektur und Funktionsweise

2.4 Modelle

Hier widmen wir ein bisschen Zeit um zu sagen, wie genau unsere Netze aussehen.

Wie haben schon gesagt, die Netze erzeugen ein Lösungsbild der Simulation unter Verwendung vom Bild vom vorherigen Zeitschritt. Da aber zwei von den Modellen auch Simulationsparameter betrachten können, ist für sie die die Netz-Eingabe ein bisschen erweitert. Die Parameter sind reelle Zahlen und die müssen mit dem Netz irgendwie passend integriert werden. Das was bei uns gut funktioniert hat, ist extra Bild-Kanäle mit konstantem Wert bei der Eingabe zu nehmen. Also ein (für die EinflieSS-Geschwindigkeit) oder zwei (Dichte und Viskosität) Felder (oder Matrizen) mit dem Wert von dem entsprechenden Parameter als extra Eingabe für das Netz. Die Ausgabe für diese Netze bleibt aber unverändert.

Kurz zu der Architektur von den Netzen. Unser Ansatz zu Bild-zu-Bild Abbildung ist auf pix2pix basiert. Pix2Pix ist ein Paper von 2015, das ein allgemeines Image-to-Image Translation Rahmenwerk vorstellt. Unsere Netze sind im Prinzip ein Versuch, dieser Ansatz im Kontext von Simulationen anzuwenden.

Der Ansatz heiSSt "Conditional Generative Adversarial Networks" und verwendetet zwei Netze – Diskriminator und Generator. Der Generator erzeugt ein Bild gegeben ein anderes als Eingabe und der Diskriminator

versucht zu raten ob das generierte Bild ein echtes Bild ist. Die Netze sind zusammen trainiert und sind so zusagen in einem Kampf miteinander. Also der Genrator soll Bilder erzeugen, die den Diskriminator austricksen müssen. Was interessant ist, ist, dass Die Loss-Funktion vom Generator den Diskriminator enthält (also wenn wir den Diskriminator als Funktion versehen) und damit selbst die Loss-Funktion wird gelernt. Das heiSSt, dass System lernt "von sich selbst" sowohl die wichtige Features von der Eingabe als auch eine passende Loss-Funktion während des Trainig-Prozesses.

Für den Generator haben wir die UNet Architektur verwendet. Wir haben es auch ResNet zum Trainieren experimentiert aber damit war der Aufwand ziemlich gröSSer und die Ergebnisse schlechter. Der Diskriminator ist derjenige, der im originalen Paper von Pix2Pix vorgeschlagen wurde und der heiSSt PatchGAN. Das ist ein Netz, das die Eingabe als Patches betrachtet und macht die Entscheidung quasi pro Patch ob dies von einem echten oder erzeugten Bild kommt.

3 Evaluation

Wenn die Modellen trainiert sind, kommt die Evaluierung. Und das erkläre ich gleich.

3.1 Strategien

Im Prinzip, hatten wir bei der Evaluierung zwei allgemeine Strategien oder Evaluierungsfälle. Diese sind individuelle Bild-Evaluierung und rekursive Evaluierung.

Individuelle Evaluierung: Bei der individuellen Evaluierung geht es um Evaluieren von einer einzigen Auswertung des Netzes. Das heiSSt, es werden ausschlieSSlich Bilder von den echten Simulationen als Eingabe verwendet und die Güte der Ausgabe wird über den gesamten Test-Set gemittelt.

Wie wir "Güte" definieren sag ich später.

Rekursive Evaluierung:

Die andere Weise auf die wir die Netze evaluieren ist rekursiv, also für ne gewisse Zahl von Schritten (oder Zeitschritten), nehmen wir die Ausgabe vom Netz wieder als Eingabe. Damit wollen wir sehen, wie der Fehler sich bei den Bildern akkumuliert und was für Artefakte können wir erwarten wenn wir das Modell für eine fast echte Simulation anwenden.

3.2 Ergebnisse

Die konkreten Metriken, die wir für die Evaluierung ausgewählt haben. Dafür haben aber zwei Sichten der Ergebnisse. Also einerseits wir vergleichen Bilder mit Bilder und zwar im Bezug auf wie ein Mensch die beiden Bilder wahrnimmt. Das heiSSt, wenn die Bilder ähnlich genug sind, betrachten wir die Ergebnisse als quasi erfolgreich. Anderseits ist unsere Aufgabe teilweise numerische. Deswegen machte es sich Sinn auch zu sagen wie weit sind "wirklich" die generierten Bilder von den echten. In diesem Fall reden wir über Metriken als mittlere und maximale Prozentuale Abweichung also objektive Differenz und nicht einfach wahrgenommene Differenz.

3.2.1 Numerische Genauigkeit

Zu diesem Aspekt der Evaluierung werde ich nicht super viel Zeit widmen. Wir werden aber die Zahlen hier in der Tabelle geben und eine kurze Bemerkung machen. Es geht um die individuelle Evaluierung und alle Zahlen sind über einige Modelle gemittelt und wir betrachten nur Modelle, die den Druckfeld verwenden. Sofort zu merken ist, die Mittlere Abweichung ist niedrig aber die Maximale hoch. Wenn wir zum. Beispiel hier auf dem Geschwindigkeit-Modell konzentrieren, die maximale Abweichung unter Verwendung vom Druckfeld ist rund 64 mal gröSSer als die mittlere. Dies hinweist, dass die Netze können im GröSSen und Ganzen die richtige Bilder erzeugen, es gibt aber jedoch gewisse Stellen oder Bereichen wo der Fehler im Bild groSS ist.

Dies ist konsistent mit diesen Beispiel-Bilder die vorhergesagt sind. Man sieht die Struktur ist die richtige, aber es gibt auch diesen Muster über den Raum. Wir haben es versucht solchen Muster mit verschiedenen Techniken zu vermeiden, also zum Beispiel die Daten zum Trainieren zu verauschen oder auf Random-Crops zu trainieren. Im groSSen Teil diese mildern den Effekt ab aber es komplett zu vermeiden in allen Fällen war es nicht möglich.

[Bilder von Artefakten]

3.2.2 Visuelle Genauigkeit

Wie gesagt aber, die wahrgenommene Qualität ist uns wichtiger und diese erlaubt uns besser die Modelle vergleichen

Unser erster Versuch diese Ähnlichkeit zu messen war mit der statistischen Korrelation zwischen den Pixelwerten. Also wenn wir alle Pixel von zwei Bildens so zu Sagen ausrollen und die beiden Reihen vergleichen, was die Korrelations dazwischen ist. Es hat sich aber herausgestellt, dass dies

uns nicht super viel Information geben kann. Bei fast allen Experimenten, war die Korrelation zwischen den vorhergesagten und den echten Bildern nahezu 1.

Wenn wir hier ein Beispiel Bild anschauen, kann man sehen was ich meine. Oben ist das Zielbild und unten ist das was ein Netz generiert hat. Die Bilder sehen fast identisch aus was die hoche Korrelation entspricht. Dafür gibts zwei Grunde. Erstens das Netz macht tatsächlich eine gute Vorhersage also die trainierte Modelle erledigen was wir wollen. Es ist aber auch war, das die Korrelation zwischen zwei nacheinander Folgenden Bilder sowieso hoch ist.

Insgesamt ergibt sich deswegen, dass wir nicht die Korrelation zum Vergleich von verschiedenen Modellen ausnutzen können.

Als eine besserer Metrik hat sich aber das PSNR gezeigt. PSNR heiSSt Peak Signal to Noise Ration und wird benutzt um die Genauigkeit von Kompressions-Algorithmen zu messen. Also zwei Bilder werden verglichen und wenn die ähnlich sind, ist der PSNR-Wert im Bereich 35-45 dB (Mess-Wert ist Dezibel). Mit dieser Metrik können wir tatsächlich die Güte von den Modellen mietender vergleichen.

[plot]

Für die Individuelle Evaluierung ist dies hier auf diesen Plots gemacht. Einige Bemerkungen dazu:

- das konstante Modell ist ziemlich inkonsistent mit den Ergebnissen. Die Verwendung des Druck-Feldes macht einen kleinen Unterschied aber nicht wirklich signifikant. Hauptsächlich beeinflusst dies die Varietät also das Modell kann besser als auch schlechter werden. Diese ist unsere Meinung nach wegen der limitierten Datenmenge für dieses Modell.

- Der Druckfeld verbessert anderseits das Geschwindigkeit-Modell. Dabei ist die PSNR ziemlich höher wenn dies mit betrachtet wird. Genau das Gegenteil ist aber beim Viskosität-Dichte Modell zu sehen. Kurz dazu, ich glaub es ist naheliegend zu vermuten, dass der Druckfeld die Komplexität des Trainieren-Problem erhöht. Unter Umständen dies kann sowohl gut als auch schlecht sein, weil in komplexeren Daten man mehrere quasi Relationen finden kann. Anderseits wenn die Daten zu komplex sind, werden die weniger modellierbar so zu sagen.

- Trotz allem, das dritte Modell hier ist gar nicht schlecht im Sinn von Bildqualität. Also zum Beispiel diese sind ein paar Bilder generiert damit und die sehen ziemlich treu zum Original.

[image]

Wenn wir die Ergebnisse bei der rekursiven Anwendung anschauen, können wir etwas Wichtiges bemerken. Diese Sind plots von der Rekursiven Anwendung von dem Geschwindigkeit-Modell für eine feste Simulation. Links ist ein Plot wo die Simulation von der ersten Index angefangen wurde und recht von Index 120. Also das 120. Bild in der Simulation wurde als "seed"-Bild für das Netz genommen. Zu Bemerken ist wie klein die Skala hier links ist und wie da gewisse Anomalie zu sehen ist. Also erstmals ist das PSNR super klein und dann ein bisschen gröSSer. Die Bilder zeigen, dass von diesen Simulationen (also start index 0) nichts benutzbar ist und man kann kaum was Verstehen.

[plot]

Es hat sich gezeigt, dass es einen groSSen Unterschied macht, welches Bild als erstes für das Netz verwendet wird. Also ob das Bild von ein bisschen nach vorne in der Simulation ist oder vom Anfang. Alle Ergebnisse (wie die hier) zeigen, dass es schwierig für die Netze ist, mit Bilder von einer sich nicht-entwickelten Simulation umzugehen. Das ganze kann viele Ursachen haben. Meine Vermutung ist aber, dass es viel weniger Bilder in den Test-Sets gibt, wo die Simulation an seiner Anfang ist. Also die Bilder zeigen in meinsten Fällen eine Simulation, die weiterentwickelt ist.

Ansonsten, die schönen Ergebnisse zeigen einen konsistenten Verlauf nach unten. Für 40 Rekursive Evaluierungen erreichen wir PSNR von 15, was schlecht genug ist um die generierten Bilder nicht zu Trauen. In fast allen Fällen senken die Warte exponentiell ab und ich hab fast keine Netze gesehen, die das negative Wachstum linear halten können und die sind eher Ausnahmen.

Hier lohnt es sich auch die ein Paar Bilder von der Evaluierung anzuschauen. Wir haben gesagt, dass diese streifige Muster ab zu da sind, die vom Netz erzeugt sind aber sind kein Teil von der Strömung. Es gibt aber auch einen andren Typ von Artefakten, die die Struktur der Strömung beeinflussen. Was ich meine, hier zum Beispiel sehen wir, dass da oben bis zum 20 Schritt nichts ungewöhnliches zu sehen ist. Danach aber kommt dieser Bereich wo das Netz erliegt etwas was nicht in der Strömung sein soll. Also, irgendwie kleine Fehler am Anfang verursachen Raschen im Bild und das Rauschen wird eventuell als Teil der Strömung vom Netz gesehen und wird dann "simuliert" quasi oder weiterverfolgt.

Bei den ersten Experimenten waren diese Fehler zimclich schlecht. Mit allen angewandten Techniken sind sie jetzt beherrschbarer aber wir haben es nicht geschafft, die komplett zu vermeiden.

3.3 Performance

Am Ende jetzt sagen auch etwas zu Performance bezüglich der Geschwindigkeit von den Modellen. Wir haben die Netze mit HiFlow verglichen und die wichtigen Ergebnissen hier zusammengefasst.

Erwartet sind die Netze in der Regel schneller. Zu bemerken ist, dass HiFlow3 ziemlich stark variieren kann, weil je nach Initialisierung vom Solver (was zufällig ist), kann es entweder lang oder kurz dauern bis eine Iteration ausgerechnet ist. Anderseits die Netze sind konsistent schnell und skallirern erstaunlich gut wenn man ein GPU benutzt. Zu bemerken ist, dass HiFlow nur über die CPU Kerne parallesiert und dementsprechend kann man nicht direkt die GPU Ergebnisse für die Netze vergleichen.

4 Schluss

Damit bin ich zum Ende. Ich bedanke mich für die Aufmerksamkeit.