

Language Modeling by Language Models

Junyan Cheng, Peter Clark, **Kyle Richardson**

May 2025

Allen Institute for Artificial Intelligence (AI2)

The big picture

- ▶ **Fully autonomous discovery**, simulate all aspects of the conventional research process (*e.g., ideation, experiment execution, paper writing*).

The big picture

- ▶ **Autonomous discovery for ML:** Discovering novel machine learning components, make our ML systems more efficient, transparent and safer..

The big picture

- ▶ **Autonomous discovery for ML:** Discovering novel machine learning components, make our ML systems more efficient, transparent and safer..

Lion optimizer (Chen et al., 2023)

Symbolic Discovery of Optimization Algorithms

Xiangning Chen^{1,2,*}, Chen Liang¹, Du Huang¹, Esteban Real¹
Kaiyuan Wang¹, Hieu Pham¹, Xunyu Dong¹, Thang Luong¹
Chao-Jui Hsieh², Yifeng Lu¹, Qiqin V. Lu¹

¹Equal & Core Contribution

¹Google ²UCLA

Abstract

We present a method to formulate algorithm discovery as program search, and apply it to discover optimization algorithms for deep neural network training. We leverage efficient search techniques to explore an infinite and sparse program space. To bridge the large generalization gap between proxy and target tasks, we also introduce program selection and simplification strategies. Our method discovers a simple and effective optimization algorithm, **Lion** (Equalized Lion Momentum). It is more memory-efficient than Adam as it only keeps track of the momentum. Different from adaptive optimizers, its update has the same magnitude for each parameter calculated through the sign operation. We compare Lion with widely used optimizers, such as Adam and Adafactor, for training a variety of models on different tasks. On image classification, Lion boosts the accuracy of ViT by up to 2% on ImageNet and saves up to 5x the pre-training compute on JFT. On vision-language contrastive learning, we achieve 88.3% zero-shot and 91.1% in-domain accuracy on ImageNet, surpassing the previous best results by 2% and 0.1%, respectively. On diffusion models, Lion outperforms Adam by achieving a better FID score and reducing the training compute by up to 2.5x. For autoregressive, masked language modeling, and fine-tuning, Lion exhibits a similar or better performance compared to Adam. Our analysis of Lion reveals that its performance gain grows with the training batch size. It also requires a smaller learning rate than Adam due to the larger norm of the update produced by the sign function. Additionally, we examine the limitations of Lion and identify scenarios where its improvements are small or not statistically significant.

The big picture

- ▶ **Autonomous discovery for ML:** Discovering novel machine learning components, make our ML systems more efficient, transparent and safer..

Lion optimizer (Chen et al., 2023)

Symbolic Discovery of Optimization Algorithms

Xiangning Chen^{1,2, *} Chen Liang¹ Du Huang² Esteban Real¹
Kaiyuan Wang¹ Hieu Pham¹ Xunyu Dong¹ Thang Luong¹
Chao-Jui Hsieh² Yifeng Lu¹ Qian V. Le¹

¹Equal & Core Contribution

¹Google ²UCLA

Abstract

We present a method to formulate algorithm discovery as program search, and apply it to discover optimization algorithms for deep neural network training. We leverage efficient search techniques to explore an infinite and sparse program space. To bridge the large generalization gap between proxy and target tasks, we also introduce program selection and simplification strategies. Our method discovers a simple and effective optimization algorithm, **Lion** (Equalized Lion Momentum). It is more memory-efficient than Adam as it only keeps track of the momentum. Different from adaptive optimizers, its update has the same magnitude for each parameter calculated through the sign operation. We compare Lion with widely used optimizers, such as Adam and Adafactor, for training a variety of models on different tasks. On image classification, Lion boosts the accuracy of ViT by up to 2% on ImageNet and saves up to 5x the pre-training compute on JFT. On vision-language contrastive learning, we achieve 88.3% zero-shot and 91.1% in-domain accuracy on ImageNet, surpassing the previous best results by 2% and 0.1%, respectively. On diffusion models, Lion outperforms Adam by achieving a better FID score and reducing the training compute by up to 2.5x. For autoregressive, masked language modeling, and fine-tuning, Lion exhibits a similar or better performance compared to Adam. Our analysis of Lion reveals that its performance gain grows with the training batch size. It also requires a smaller learning rate than Adam due to the larger norm of the update produced by the sign function. Additionally, we examine the limitations of Lion and identify scenarios where its improvements are small or not statistically significant.

Narrow (no LLMs), clear goals.

The big picture

- **Autonomous discovery for ML:** Discovering novel machine learning components, make our ML systems more efficient, transparent and safer..

Lion optimizer (Chen et al., 2023)

Symbolic Discovery of Optimization Algorithms

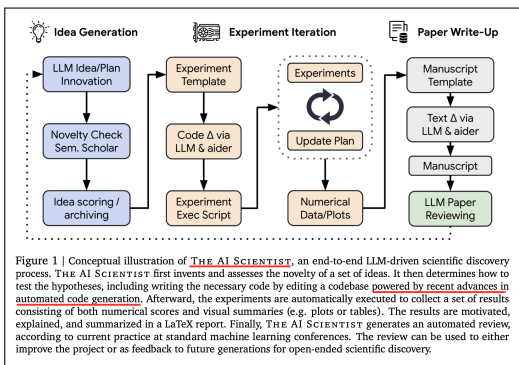
Xiangning Chen^{1,2, *} Chen Liang¹ Du Huang² Esteban Real¹
Kaiyuan Wang¹ Hieu Pham¹ Xinyi Dong¹ Thang Luong¹
Cho-Jui Hsieh² Yifeng Lu¹ Qiuqiang Ye¹

¹Equal & Core Contribution
¹Google ²UCLA

Abstract

We present a method to formulate algorithm discovery as program search, and apply it to discover optimization algorithms for deep neural network training. We leverage efficient search techniques to explore an infinite and sparse program space. To bridge the large generalization gap between proxy and target tasks, we also introduce program selection and simplification strategies. Our method discovers a simple and effective optimization algorithm, *Lion* (*Roofed Lion Momentum*). It is more memory-efficient than Adam as it only keeps track of the momentum. Different from adaptive optimizers, its update has the same magnitude for each parameter calculated through the sign operation. We compare Lion with widely used optimizers, such as Adam and Adafactor, for training a variety of models on different tasks. On image classification, Lion boosts the accuracy of ViT by up to 2% on ImageNet and saves up to 5x the pre-training compute on JFT. On vision-language contrastive learning, we achieve 88.3% zero-shot and 91.1% finetuning accuracy on ImageNet, surpassing the previous best results by 2% and 0.1%, respectively. On diffusion models, Lion outperforms Adam by achieving a better FID score and reducing the training compute by up to 2.5x. For autoregressive, masked language modeling, and fine-tuning, Lion exhibits a similar or better performance compared to Adam. Our analysis of Lion reveals that its performance gain grows with the training batch size. It also requires a smaller learning rate than Adam due to the larger norm of the update produced by the sign function. Additionally, we examine the limitations of Lion and identify scenarios where its improvements are small or not statistically significant.

AI Scientist (Lu et al., 2024)



Narrow (no LLMs), clear goals.

The big picture

- **Autonomous discovery for ML:** Discovering novel machine learning components, make our ML systems more efficient, transparent and safer..

Lion optimizer (Chen et al., 2023)

Symbolic Discovery of Optimization Algorithms

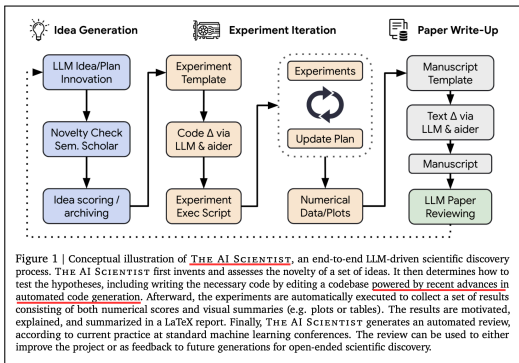
Xiangning Chen^{1,2}, Chen Liang¹, Du Huang², Esteban Real¹
Kaiyuan Wang¹, Hieu Pham¹, Xinyi Dong¹, Thang Luong¹
Cho-Jui Hsieh², Yifeng Lu¹, Qiuqiang Ye¹

¹Equal & Core Contribution
²Google

Abstract

We present a method to formulate algorithm discovery as program search, and apply it to discover optimization algorithms for deep neural network training. We leverage efficient search techniques to explore an infinite and sparse program space. To bridge the large generalization gap between proxy and target tasks, we also introduce program selection and simplification strategies. Our method discovers a simple and effective optimization algorithm, *Lion* (see [Lion GitHub repository](#)). It is more memory-efficient than Adam as it only keeps track of the momentum. Different from adaptive optimizers, its update has the same magnitude for each parameter calculated through the sign operation. We compare Lion with widely used optimizers, such as Adam and Adafactor, for training a variety of models on different tasks. On image classification, Lion boosts the accuracy of ViT by up to 2% on ImageNet and saves up to 5x the pre-training compute on JFT. On vision-language contrastive learning, we achieve 88.3% zero-shot and 91.1% in-domain accuracy on ImageNet, surpassing the previous best results by 2% and 0.1%, respectively. On diffusion models, Lion outperforms Adam by achieving a better FID score and reducing the training compute by up to 2.5x. For autoregressive, masked language modeling, and fine-tuning, Lion exhibits a similar or better performance compared to Adam. Our analysis of Lion reveals that its performance gain grows with the training batch size. It also requires a smaller learning rate than Adam due to the larger norm of the update produced by the sign function. Additionally, we examine the limitations of Lion and identify scenarios where its improvements are small or not statistically significant.

AI Scientist (Lu et al., 2024)



Narrow (no LLMs), clear goals.

Broad (LLM-driven), **unclear** goals.

The big picture

Lion optimizer (Chen et al., 2023)

Symbolic Discovery of Optimization Algorithms

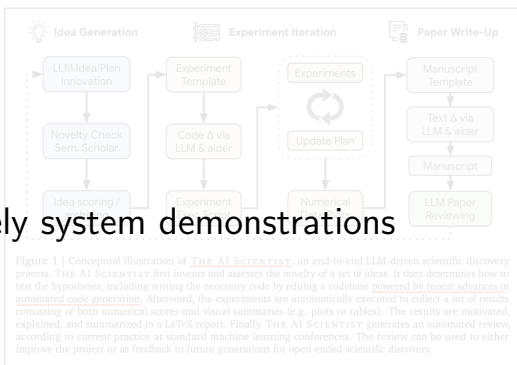
Xiangting Chen^{1,2,*} Chen Liang¹ De Huang² Fuzhan Bao²
Kaiyuan Wang² Hien Pham² Xiangyi Dong² Shang Luong²
Chao-Jui Hsieh² Yifeng Lu² Quan V. Le²

¹Equal & Core Contribution
²Google ³UCLA

Abstract

We present a method to formulate algorithm discovery as program search, and apply it to the optimization domain. We leverage off-the-shelf deep neural network trunks to bridge algorithm discovery and deep learning, and introduce a novel search algorithm to discover novel optimization algorithms. Our method achieves state-of-the-art performance on various optimization tasks. It is more memory-efficient than Adam as it only keeps track of the momentum. Different from adaptive optimizers, Lion uses the same magnitude for each parameter calculated through the sign operation. We compare Lion with widely used optimizers, such as Adam and Adafactor, for training a variety of models on different tasks. On image classification, Lion beats the accuracy of ViT by up to 2% on ImageNet and saves up to 5% the pre-training compute on DPT. On vision-language contrastive learning, we achieve 60.3% zero-shot and 91.1% in-domain accuracy on ImageNet, surpassing the previous best results by 2% and 0.1%, respectively. On diffusion models, Lion outperforms Adam by achieving a better FID score and reducing the training compute by up to 2.3x. For autoregressive, masked language modeling, and fine-tuning, Lion exhibits a similar or better performance compared to Adam. Our analysis of Lion reveals that its performance gain grows with the training batch size. It also requires a smaller learning rate than Adam due to the larger norm of the update produced by the sign function. Additionally, we examine the limitations of Lion and identify scenarios where its improvements are small or not statistically significant.

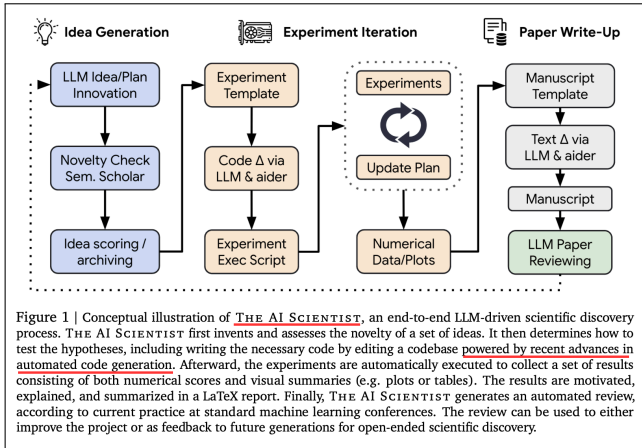
AI Scientist (Lu et al., 2024)



Narrow (no LLMs), clear goals.

Broad (LLM-driven), unclear goals.

1. Tasks: What are the target discovery tasks?



- What tasks and discovery problems should we be working on to make progress? Community has not yet come up with clear tasks or metrics.

1. Tasks: What are the target discovery tasks?

Lion optimizer (Chen et al., 2023)

Symbolic Discovery of Optimization Algorithms

Xiangning Chen^{1,2,1} Chen Liang¹ Du Huang¹ Eotchen Reu¹
Kaiyuan Wang¹ Hieu Phan¹ Xuanqi Dong¹ Thang Luong¹
Chao-Jui Hsieh² Yifeng Lu¹ Quoc V. Le¹

¹Equal & Core Contribution
¹Google ²UCLA

Abstract

We present a method to formulate algorithm discovery as program search, and apply it to discover optimization algorithms for deep neural network training. We leverage efficient search techniques to explore an infinite and sparse program space. To bridge the large generalization gap between proxy and target tasks, we also introduce program selection and simplification strategies. Our method discovers a simple and effective optimization algorithm, Lion (GoofyCool Slim Momentum). It is more memory-efficient than Adam as it only keeps track of the momentum. Different from adaptive optimizers, its update has the same magnitude for each parameter calculated through the sign operation. We compare Lion with widely used optimizers, such as Adam and Adafactor, for training a variety of models on different tasks. On image classification, Lion boosts the accuracy of ViT by up to 2% on ImageNet and saves up to 5x the pre-training compute on JFT. On vision-language contrastive learning, we achieve 88.3% zero-shot and 91.1% few-shot accuracy on ImageNet, surpassing the previous best results by 2% and 0.1%, respectively. On diffusion models, Lion outperforms Adam by achieving a better FID score and reducing the training compute by up to 2.5x. For autoregressive, masked language modeling, and fine-tuning, Lion exhibits a similar or better performance compared to Adam. Our analysis of Lion reveals that its performance gain grows with the training batch size. It also requires a smaller learning rate than Adam due to the larger norm of the update produced by the sign function. Additionally, we examine the limitations of Lion and identify scenarios where its improvements are small or not statistically significant.

AI Scientist (Lu et al., 2024)

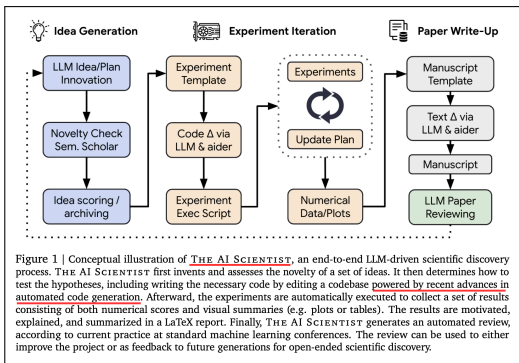


Figure 1 | Conceptual illustration of **THE AI SCIENTIST**, an end-to-end LLM-driven scientific discovery process. **THE AI SCIENTIST** first invents and assesses the novelty of a set of ideas. It then determines how to test the hypotheses, including writing the necessary code by editing a codebase powered by recent advances in automated code generation. Afterward, the experiments are automatically executed to collect a set of results consisting of both numerical scores and visual summaries (e.g. plots or tables). The results are motivated, explained, and summarized in a LaTeX report. Finally, **THE AI SCIENTIST** generates an automated review, according to current practice at standard machine learning conferences. The review can be used to either improve the project or as feedback to future generations for open-ended scientific discovery.

Narrow (no LLMs), clear goals.

Broad (LLM-driven), unclear goals.

1. Tasks: What are the target discovery tasks?

Lion optimizer (Chen et al., 2023)

Symbolic Discovery of Optimization Algorithms

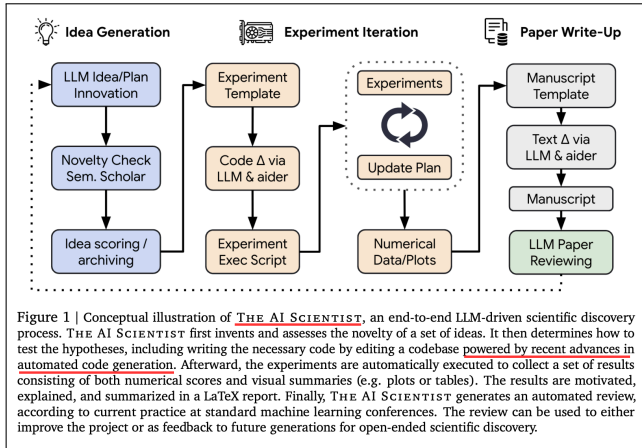
Xiangning Chen^{1,2,1} Chen Liang¹ Du Huang¹ Eotchen Reu¹
Kaiyuan Wang¹ Hieu Phan¹ Xuanqi Dong¹ Thang Luong¹
Chao-Jui Hsieh² Yifeng Lu¹ Quoc V. Le¹

¹Equal & Core Contribution
¹Google ²UCLA

Abstract

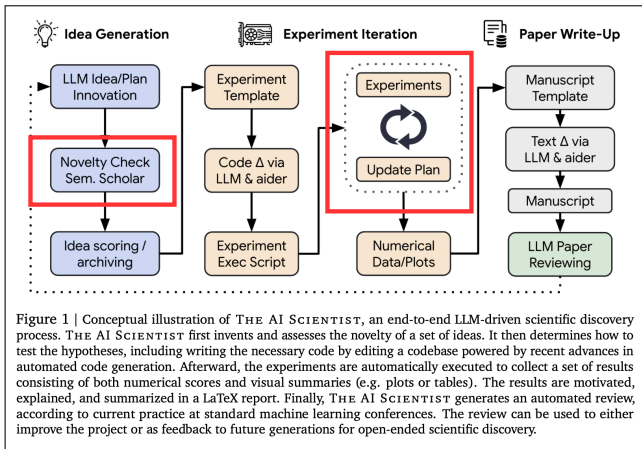
We present a method to formulate algorithm discovery as program search, and apply it to discover optimization algorithms for deep neural network training. We leverage efficient search techniques to explore an infinite and sparse program space. To bridge the large generalization gap between proxy and target tasks, we also introduce program selection and simplification strategies. Our method discovers a simple and effective optimization algorithm, Lion (Go/Go2/Go3/Go4/Go5/Go6/Go7/Go8/Go9/Go10/Go11/Go12/Go13/Go14/Go15/Go16/Go17/Go18/Go19/Go20/Go21/Go22/Go23/Go24/Go25/Go26/Go27/Go28/Go29/Go30/Go31/Go32/Go33/Go34/Go35/Go36/Go37/Go38/Go39/Go40/Go41/Go42/Go43/Go44/Go45/Go46/Go47/Go48/Go49/Go50/Go51/Go52/Go53/Go54/Go55/Go56/Go57/Go58/Go59/Go60/Go61/Go62/Go63/Go64/Go65/Go66/Go67/Go68/Go69/Go70/Go71/Go72/Go73/Go74/Go75/Go76/Go77/Go78/Go79/Go80/Go81/Go82/Go83/Go84/Go85/Go86/Go87/Go88/Go89/Go90/Go91/Go92/Go93/Go94/Go95/Go96/Go97/Go98/Go99/Go100/Go101/Go102/Go103/Go104/Go105/Go106/Go107/Go108/Go109/Go110/Go111/Go112/Go113/Go114/Go115/Go116/Go117/Go118/Go119/Go120/Go121/Go122/Go123/Go124/Go125/Go126/Go127/Go128/Go129/Go130/Go131/Go132/Go133/Go134/Go135/Go136/Go137/Go138/Go139/Go140/Go141/Go142/Go143/Go144/Go145/Go146/Go147/Go148/Go149/Go150/Go151/Go152/Go153/Go154/Go155/Go156/Go157/Go158/Go159/Go160/Go161/Go162/Go163/Go164/Go165/Go166/Go167/Go168/Go169/Go170/Go171/Go172/Go173/Go174/Go175/Go176/Go177/Go178/Go179/Go180/Go181/Go182/Go183/Go184/Go185/Go186/Go187/Go188/Go189/Go190/Go191/Go192/Go193/Go194/Go195/Go196/Go197/Go198/Go199/Go200/Go201/Go202/Go203/Go204/Go205/Go206/Go207/Go208/Go209/Go210/Go211/Go212/Go213/Go214/Go215/Go216/Go217/Go218/Go219/Go220/Go221/Go222/Go223/Go224/Go225/Go226/Go227/Go228/Go229/Go230/Go231/Go232/Go233/Go234/Go235/Go236/Go237/Go238/Go239/Go240/Go241/Go242/Go243/Go244/Go245/Go246/Go247/Go248/Go249/Go250/Go251/Go252/Go253/Go254/Go255/Go256/Go257/Go258/Go259/Go260/Go261/Go262/Go263/Go264/Go265/Go266/Go267/Go268/Go269/Go270/Go271/Go272/Go273/Go274/Go275/Go276/Go277/Go278/Go279/Go280/Go281/Go282/Go283/Go284/Go285/Go286/Go287/Go288/Go289/Go290/Go291/Go292/Go293/Go294/Go295/Go296/Go297/Go298/Go299/Go300/Go301/Go302/Go303/Go304/Go305/Go306/Go307/Go308/Go309/Go310/Go311/Go312/Go313/Go314/Go315/Go316/Go317/Go318/Go319/Go320/Go321/Go322/Go323/Go324/Go325/Go326/Go327/Go328/Go329/Go330/Go331/Go332/Go333/Go334/Go335/Go336/Go337/Go338/Go339/Go340/Go341/Go342/Go343/Go344/Go345/Go346/Go347/Go348/Go349/Go350/Go351/Go352/Go353/Go354/Go355/Go356/Go357/Go358/Go359/Go360/Go361/Go362/Go363/Go364/Go365/Go366/Go367/Go368/Go369/Go370/Go371/Go372/Go373/Go374/Go375/Go376/Go377/Go378/Go379/Go380/Go381/Go382/Go383/Go384/Go385/Go386/Go387/Go388/Go389/Go390/Go391/Go392/Go393/Go394/Go395/Go396/Go397/Go398/Go399/Go400/Go401/Go402/Go403/Go404/Go405/Go406/Go407/Go408/Go409/Go410/Go411/Go412/Go413/Go414/Go415/Go416/Go417/Go418/Go419/Go420/Go421/Go422/Go423/Go424/Go425/Go426/Go427/Go428/Go429/Go430/Go431/Go432/Go433/Go434/Go435/Go436/Go437/Go438/Go439/Go440/Go441/Go442/Go443/Go444/Go445/Go446/Go447/Go448/Go449/Go450/Go451/Go452/Go453/Go454/Go455/Go456/Go457/Go458/Go459/Go460/Go461/Go462/Go463/Go464/Go465/Go466/Go467/Go468/Go469/Go470/Go471/Go472/Go473/Go474/Go475/Go476/Go477/Go478/Go479/Go480/Go481/Go482/Go483/Go484/Go485/Go486/Go487/Go488/Go489/Go490/Go491/Go492/Go493/Go494/Go495/Go496/Go497/Go498/Go499/Go500/Go501/Go502/Go503/Go504/Go505/Go506/Go507/Go508/Go509/Go510/Go511/Go512/Go513/Go514/Go515/Go516/Go517/Go518/Go519/Go520/Go521/Go522/Go523/Go524/Go525/Go526/Go527/Go528/Go529/Go530/Go531/Go532/Go533/Go534/Go535/Go536/Go537/Go538/Go539/Go540/Go541/Go542/Go543/Go544/Go545/Go546/Go547/Go548/Go549/Go550/Go551/Go552/Go553/Go554/Go555/Go556/Go557/Go558/Go559/Go560/Go561/Go562/Go563/Go564/Go565/Go566/Go567/Go568/Go569/Go570/Go571/Go572/Go573/Go574/Go575/Go576/Go577/Go578/Go579/Go580/Go581/Go582/Go583/Go584/Go585/Go586/Go587/Go588/Go589/Go590/Go591/Go592/Go593/Go594/Go595/Go596/Go597/Go598/Go599/Go600/Go601/Go602/Go603/Go604/Go605/Go606/Go607/Go608/Go609/Go610/Go611/Go612/Go613/Go614/Go615/Go616/Go617/Go618/Go619/Go620/Go621/Go622/Go623/Go624/Go625/Go626/Go627/Go628/Go629/Go630/Go631/Go632/Go633/Go634/Go635/Go636/Go637/Go638/Go639/Go640/Go641/Go642/Go643/Go644/Go645/Go646/Go647/Go648/Go649/Go650/Go651/Go652/Go653/Go654/Go655/Go656/Go657/Go658/Go659/Go660/Go661/Go662/Go663/Go664/Go665/Go666/Go667/Go668/Go669/Go670/Go671/Go672/Go673/Go674/Go675/Go676/Go677/Go678/Go679/Go680/Go681/Go682/Go683/Go684/Go685/Go686/Go687/Go688/Go689/Go690/Go691/Go692/Go693/Go694/Go695/Go696/Go697/Go698/Go699/Go700/Go701/Go702/Go703/Go704/Go705/Go706/Go707/Go708/Go709/Go710/Go711/Go712/Go713/Go714/Go715/Go716/Go717/Go718/Go719/Go720/Go721/Go722/Go723/Go724/Go725/Go726/Go727/Go728/Go729/Go730/Go731/Go732/Go733/Go734/Go735/Go736/Go737/Go738/Go739/Go740/Go741/Go742/Go743/Go744/Go745/Go746/Go747/Go748/Go749/Go750/Go751/Go752/Go753/Go754/Go755/Go756/Go757/Go758/Go759/Go760/Go761/Go762/Go763/Go764/Go765/Go766/Go767/Go768/Go769/Go770/Go771/Go772/Go773/Go774/Go775/Go776/Go777/Go778/Go779/Go780/Go781/Go782/Go783/Go784/Go785/Go786/Go787/Go788/Go789/Go790/Go791/Go792/Go793/Go794/Go795/Go796/Go797/Go798/Go799/Go800/Go801/Go802/Go803/Go804/Go805/Go806/Go807/Go808/Go809/Go810/Go811/Go812/Go813/Go814/Go815/Go816/Go817/Go818/Go819/Go820/Go821/Go822/Go823/Go824/Go825/Go826/Go827/Go828/Go829/Go830/Go831/Go832/Go833/Go834/Go835/Go836/Go837/Go838/Go839/Go840/Go841/Go842/Go843/Go844/Go845/Go846/Go847/Go848/Go849/Go850/Go851/Go852/Go853/Go854/Go855/Go856/Go857/Go858/Go859/Go860/Go861/Go862/Go863/Go864/Go865/Go866/Go867/Go868/Go869/Go870/Go871/Go872/Go873/Go874/Go875/Go876/Go877/Go878/Go879/Go880/Go881/Go882/Go883/Go884/Go885/Go886/Go887/Go888/Go889/Go890/Go891/Go892/Go893/Go894/Go895/Go896/Go897/Go898/Go899/Go900/Go901/Go902/Go903/Go904/Go905/Go906/Go907/Go908/Go909/Go910/Go911/Go912/Go913/Go914/Go915/Go916/Go917/Go918/Go919/Go920/Go921/Go922/Go923/Go924/Go925/Go926/Go927/Go928/Go929/Go930/Go931/Go932/Go933/Go934/Go935/Go936/Go937/Go938/Go939/Go940/Go941/Go942/Go943/Go944/Go945/Go946/Go947/Go948/Go949/Go950/Go951/Go952/Go953/Go954/Go955/Go956/Go957/Go958/Go959/Go960/Go961/Go962/Go963/Go964/Go965/Go966/Go967/Go968/Go969/Go970/Go971/Go972/Go973/Go974/Go975/Go976/Go977/Go978/Go979/Go980/Go981/Go982/Go983/Go984/Go985/Go986/Go987/Go988/Go989/Go990/Go991/Go992/Go993/Go994/Go995/Go996/Go997/Go998/Go999/Go1000/Go1001/Go1002/Go1003/Go1004/Go1005/Go1006/Go1007/Go1008/Go1009/Go1010/Go1011/Go1012/Go1013/Go1014/Go1015/Go1016/Go1017/Go1018/Go1019/Go1020/Go1021/Go1022/Go1023/Go1024/Go1025/Go1026/Go1027/Go1028/Go1029/Go1030/Go1031/Go1032/Go1033/Go1034/Go1035/Go1036/Go1037/Go1038/Go1039/Go1040/Go1041/Go1042/Go1043/Go1044/Go1045/Go1046/Go1047/Go1048/Go1049/Go1050/Go1051/Go1052/Go1053/Go1054/Go1055/Go1056/Go1057/Go1058/Go1059/Go1060/Go1061/Go1062/Go1063/Go1064/Go1065/Go1066/Go1067/Go1068/Go1069/Go1070/Go1071/Go1072/Go1073/Go1074/Go1075/Go1076/Go1077/Go1078/Go1079/Go1080/Go1081/Go1082/Go1083/Go1084/Go1085/Go1086/Go1087/Go1088/Go1089/Go1090/Go1091/Go1092/Go1093/Go1094/Go1095/Go1096/Go1097/Go1098/Go1099/Go1100/Go1101/Go1102/Go1103/Go1104/Go1105/Go1106/Go1107/Go1108/Go1109/Go1110/Go1111/Go1112/Go1113/Go1114/Go1115/Go1116/Go1117/Go1118/Go1119/Go1120/Go1121/Go1122/Go1123/Go1124/Go1125/Go1126/Go1127/Go1128/Go1129/Go1130/Go1131/Go1132/Go1133/Go1134/Go1135/Go1136/Go1137/Go1138/Go1139/Go1140/Go1141/Go1142/Go1143/Go1144/Go1145/Go1146/Go1147/Go1148/Go1149/Go1150/Go1151/Go1152/Go1153/Go1154/Go1155/Go1156/Go1157/Go1158/Go1159/Go1160/Go1161/Go1162/Go1163/Go1164/Go1165/Go1166/Go1167/Go1168/Go1169/Go1170/Go1171/Go1172/Go1173/Go1174/Go1175/Go1176/Go1177/Go1178/Go1179/Go1180/Go1181/Go1182/Go1183/Go1184/Go1185/Go1186/Go1187/Go1188/Go1189/Go1190/Go1191/Go1192/Go1193/Go1194/Go1195/Go1196/Go1197/Go1198/Go1199/Go1200/Go1201/Go1202/Go1203/Go1204/Go1205/Go1206/Go1207/Go1208/Go1209/Go1210/Go1211/Go1212/Go1213/Go1214/Go1215/Go1216/Go1217/Go1218/Go1219/Go1220/Go1221/Go1222/Go1223/Go1224/Go1225/Go1226/Go1227/Go1228/Go1229/Go1230/Go1231/Go1232/Go1233/Go1234/Go1235/Go1236/Go1237/Go1238/Go1239/Go1240/Go1241/Go1242/Go1243/Go1244/Go1245/Go1246/Go1247/Go1248/Go1249/Go1250/Go1251/Go1252/Go1253/Go1254/Go1255/Go1256/Go1257/Go1258/Go1259/Go1260/Go1261/Go1262/Go1263/Go1264/Go1265/Go1266/Go1267/Go1268/Go1269/Go1270/Go1271/Go1272/Go1273/Go1274/Go1275/Go1276/Go1277/Go1278/Go1279/Go1280/Go1281/Go1282/Go1283/Go1284/Go1285/Go1286/Go1287/Go1288/Go1289/Go1290/Go1291/Go1292/Go1293/Go1294/Go1295/Go1296/Go1297/Go1298/Go1299/Go1300/Go1301/Go1302/Go1303/Go1304/Go1305/Go1306/Go1307/Go1308/Go1309/Go1310/Go1311/Go1312/Go1313/Go1314/Go1315/Go1316/Go1317/Go1318/Go1319/Go1320/Go1321/Go1322/Go1323/Go1324/Go1325/Go1326/Go1327/Go1328/Go1329/Go1330/Go1331/Go1332/Go1333/Go1334/Go1335/Go1336/Go1337/Go1338/Go1339/Go1340/Go1341/Go1342/Go1343/Go1344/Go1345/Go1346/Go1347/Go1348/Go1349/Go1350/Go1351/Go1352/Go1353/Go1354/Go1355/Go1356/Go1357/Go1358/Go1359/Go1360/Go1361/Go1362/Go1363/Go1364/Go1365/Go1366/Go1367/Go1368/Go1369/Go1370/Go1371/Go1372/Go1373/Go1374/Go1375/Go1376/Go1377/Go1378/Go1379/Go1380/Go1381/Go1382/Go1383/Go1384/Go1385/Go1386/Go1387/Go1388/Go1389/Go1390/Go1391/Go1392/Go1393/Go1394/Go1395/Go1396/Go1397/Go1398/Go1399/Go1400/Go1401/Go1402/Go1403/Go1404/Go1405/Go1406/Go1407/Go1408/Go1409/Go1410/Go1411/Go1412/Go1413/Go1414/Go1415/Go1416/Go1417/Go1418/Go1419/Go1420/Go1421/Go1422/Go1423/Go1424/Go1425/Go1426/Go1427/Go1428/Go1429/Go1430/Go1431/Go1432/Go1433/Go1434/Go1435/Go1436/Go1437/Go1438/Go1439/Go1440/Go1441/Go1442/Go1443/Go1444/Go1445/Go1446/Go1447/Go1448/Go1449/Go1450/Go1451/Go1452/Go1453/Go1454/Go1455/Go1456/Go1457/Go1458/Go1459/Go1460/Go1461/Go1462/Go1463/Go1464/Go1465/Go1466/Go1467/Go1468/Go1469/Go1470/Go1471/Go1472/Go1473/Go1474/Go1475/Go1476/Go1477/Go1478/Go1479/Go1480/Go1481/Go1482/Go1483/Go1484/Go1485/Go1486/Go1487/Go1488/Go1489/Go1490/Go1491/Go1492/Go1493/Go1494/Go1495/Go1496/Go1497/Go1498/Go1499/Go1500/Go1501/Go1502/Go1503/Go1504/Go1505/Go1506/Go1507/Go1508/Go1509/Go1510/Go1511/Go1512/Go1513/Go1514/Go1515/Go1516/Go1517/Go1518/Go1519/Go1520/Go1521/Go1522/Go1523/Go1524/Go1525/Go1526/Go1527/Go1528/Go1529/Go1530/Go1531/Go1532/Go1533/Go1534/Go1535/Go1536/Go1537/Go1538/Go1539/Go1540/Go1541/Go1542/Go1543/Go1544/Go1545/Go1546/Go1547/Go1548/Go1549/Go1550/Go1551/Go1552/Go1553/Go1554/Go1555/Go1556/Go1557/Go1558/Go1559/Go1560/Go1561/Go1562/Go1563/Go1564/Go1565/Go1566/Go1567/Go1568/Go1569/Go1570/Go1571/Go1572/Go1573/Go1574/Go1575/Go1576/Go1577/Go1578/Go1579/Go1580/Go1581/Go1582/Go1583/Go1584/Go1585/Go1586/Go1587/Go1588/Go1589/Go1590/Go1591/Go1592/Go1593/Go1594/Go1595/Go1596/Go1597/Go1598/Go1599/Go1600/Go1601/Go1602/Go1603/Go1604/Go1605/Go1606/Go1607/Go1608/Go1609/Go1610/Go1611/Go1612/Go1613/Go1614/Go1615/Go1616/Go1617/Go1618/Go1619/Go1620/Go1621/Go1622/Go1623/Go1624/Go1625/Go1626/Go1627/Go1628/Go1629/Go1630/Go1631/Go1632/Go1633/Go1634/Go1635/Go1636/Go1637/Go1638/Go1639/Go1640/Go1641/Go1642/Go1643/Go1644/Go1645/Go1646/Go1647/Go1648/Go1649/Go1650/Go1651/Go1652/Go1653/Go1654/Go1655/Go1656/Go1657/Go1658/Go1659/Go1660/Go1661/Go1662/Go1663/Go1664/Go1665/Go1666/Go1667/Go1668/Go1669/Go1670/Go1671/Go1672/Go1673/Go1674/Go1675/Go1676/Go1677/Go1678/Go1679/Go1680/Go1681/Go1682/Go1683/Go1684/Go1685/Go1686/Go1687/Go1688/Go1689/Go1690/Go1691/Go1692/Go1693/Go1694/Go1695/Go1696/Go1697/Go1698/Go1699/Go1700/Go1701/Go1702/Go1703/Go1704/Go1705/Go1706/Go1707/Go1708/Go1709/Go1710/Go1711/Go1712/Go1713/Go1714/Go1715/Go1716/Go1717/Go1718/Go1719/Go1720/Go1721/Go1722/Go1723/Go1724/Go1725/Go1726/Go1727/Go1728/Go1729/Go1730/Go1731/Go1732/Go1733/Go1734/Go1735/Go1736/Go1737/Go1738/Go1739/Go1740/Go1741/Go1742/Go1743/Go1744/Go1745/Go1746/Go1747/Go1748/Go1749/Go1750/Go1751/Go1752/Go1753/Go1754/Go1755/Go1756/Go1757/Go1758/Go1759/Go1760/Go1761/Go1762/Go1763/Go1764/Go1765/Go1766/Go1767/Go1768/Go1769/Go1770/Go1771/Go1772/Go1773/Go1774/Go1775/Go1776/Go1777/Go1778/Go1779/Go1780/Go1781/Go1782/Go1783/Go1784/Go1785/Go1786/Go1787/Go1788/Go1789/Go1790/Go1791/Go1792/Go1793/Go1794/Go1795/Go1796/Go1797/Go1798/Go1799/Go1800/Go1801/Go1802/Go1803/Go1804/Go1805/Go1806/Go1807/Go1808/Go1809/Go1810/Go1811/Go1812/Go1813/Go1814/Go1815/Go1816/Go1817/Go1818/Go1819/Go1820/Go1821/Go1822/Go1823/Go1824/Go1825/Go1826/Go1827/Go1828/Go1829/Go1830/Go1831/Go1832/Go1833/Go1834/Go1835/Go1836/Go1837/Go1838/Go1839/Go1840/Go1841/Go1842/Go1843/Go1844/Go1845/Go1846/Go1847/Go1848/Go1849/Go1850/Go1851/Go1852/Go1853/Go1854/Go1855/Go1856/Go1857/Go1858/Go1859/Go1860/Go1861/Go1862/Go1863/Go1864/Go1865/Go1866/Go1867/Go1868/Go1869/Go1870/Go1871/Go1872/Go1873/Go1874/Go1875/Go1876/Go1877/Go1878/Go1879/Go1880/Go1881/Go1882/Go1883/Go1884/Go1885/Go1886/Go1887/Go1888/Go1889/Go1890/Go1891/Go1892/Go1893/Go1894/Go1895/Go1896/Go1897/Go1898/Go1899/Go1900/Go1901/Go1902/Go1903/Go1904/Go1905/Go1906/Go1907/Go1908/Go1909/Go1910/Go1911/Go1912/Go1913/Go1914/Go1915/Go1916/Go1917/Go1918/Go1919/Go1920/Go1921/Go1922/Go1923/Go1924/Go1925/Go1926/Go1927/Go1928/Go1929/Go1930/Go1931/Go1932/Go1933/Go1934/Go1935/Go1936/Go1937/Go1938/Go1939/Go1940/Go1941/Go1942/Go1943/Go1944/Go1945/Go1946/Go1947/Go1948/Go1949/Go1950/Go1951/Go1952/Go1953/Go1954/Go1955/Go1956/Go1957/Go1958/Go1959/Go1960/Go1961/Go1962/Go1963/Go1964/Go1965/Go1966/Go1967/Go1968/Go1969/Go1970/Go1971/Go1972/Go1973/Go1974/Go1975/Go1976/Go1977/Go1978/Go1979/Go1980/Go1981/Go1982/Go1983/Go1984/Go1985/Go1986/Go1987/Go1988/Go1989/Go1990/Go1991/Go1992/Go1993/Go1994/Go1995/Go1996/Go1997/Go1998/Go1999/Go2000/Go2001/Go2002/Go2003/Go2004/Go2005/Go2006/Go2007/Go2008/Go2009/Go2010/Go2011/Go2012/Go2013/Go2014/Go2015/Go2016/Go2017/Go2018/Go2019/Go2020/Go2021/Go2022/Go2023/Go2024/Go2025/Go2026/Go2027/Go2028/Go2029/Go2030/Go2031/Go2032/Go2033/Go2034/Go2035/Go2036/Go2037/Go2038/Go2039/Go2040/Go2041/Go2042/Go2043/Go2044/Go2045/Go2046/Go2047/Go2048/Go2049/Go2050/Go2051/Go2052/Go2053/Go2054/Go2055/Go2056/Go2057/Go2058/Go2059/Go2060/Go2061/Go2062/Go2063/Go2064/Go2065/Go2066/Go2067/Go2068/Go2069/Go2070/Go2071/Go2072/Go2073/Go2074/Go2075/Go2076/Go2077/Go2078/Go2079/Go2080/Go2081/Go2082/Go2083/Go2084/Go2085/Go2086/Go2087/Go2088/Go2089/Go2090/Go2091/Go2092/Go2093/Go2094/Go2095/Go2096/Go2097/Go2098/Go2099/Go2100/Go2101/Go2102/Go2103/Go2104/Go2105/Go2106/Go2107/Go2108/Go2109/Go2110/Go2111/Go2112/Go2113/Go2114/Go2115/Go2116/Go2117/Go2118/Go2119/Go2120/Go2121/Go2122/Go2123/Go2124/Go2125/Go2126/Go2127/Go2128/Go2129/Go2130/Go2131/Go2132/Go2133/Go2134/Go2135/Go2136/Go2137/Go2138/Go2139/Go2140/Go2141/Go2142/Go2143/Go2144/Go2145/Go2146/Go2147/Go2148/Go2149/Go2150/Go2151/Go2152/Go2153/Go2154/Go2155/Go2156/Go2157/Go2158/Go2159/Go2160/Go2161/Go2162/Go2163/Go2164/Go2165/Go2166/Go2167/Go2168/Go2169/Go2170/Go2171/Go2172/Go2173/Go2174/Go2175/Go2176/Go2177/Go2178/Go2179/Go2180/Go2181/Go2182/Go218

2. System design: How should we build such systems?

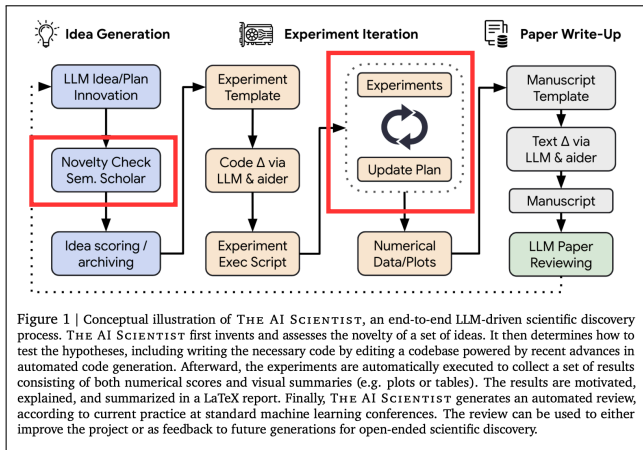


- Do these discovery workflows make sense? What are their limits? Should be efficient, cost-effective, transparent.

2. System design: How should we build such systems?

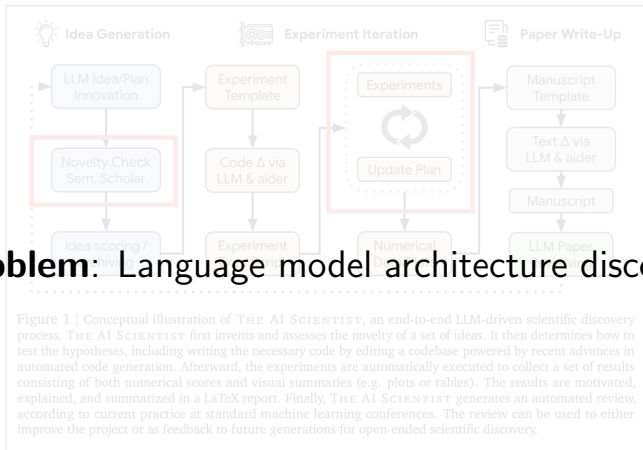


2. System design: How should we build such systems?



Proposed a new algorithmic framework for discovery, allows us to address technical issues, devise generalized algorithms.

2. System design: How should we build such systems?



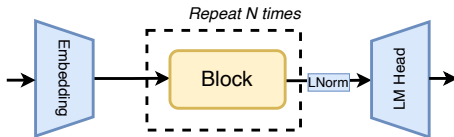
Proposed a new algorithmic framework for discovery, allows us to address technical issues, devise generalized algorithms.

Language model architecture design discovery: what?

- ▶ Finding improved layer designs for autoregressive language models.

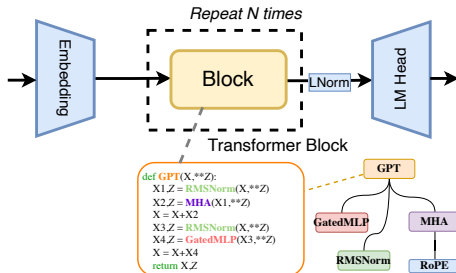
Language model architecture design discovery: what?

- Finding improved layer designs for autoregressive language models.



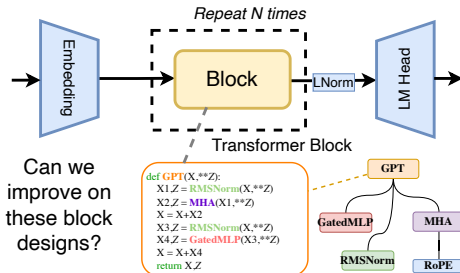
Language model architecture design discovery: what?

- Finding improved layer designs for autoregressive language models.



Language model architecture design discovery: what?

- Finding improved layer designs for autoregressive language models.



At its core, a **code discovery** problem, similar goals to AutoML and Neural architecture search (**NAS**), model full research pipeline.

Why is this an interesting problem?




IMAGE CREDITS: NATEE127 / GETTY IMAGES

TechCrunch

Latest Startups Venture Apple Security AI Apps Startup Battlefield Events Podcasts Newsletters

Sign In

AI

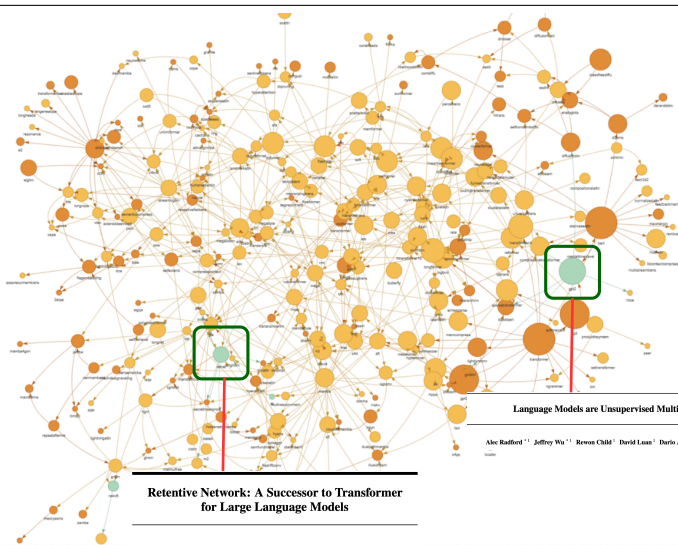
TTT models might be the next frontier in generative AI

Kyle Wiggers · 1:47 PM PDT · July 17, 2024

After years of dominance by the form of AI known as the transformer, the hunt is on for new architectures.

Transformers underpin OpenAI's video-generating model Sora, and they're at the heart of text-generating models like Anthropic's Claude, Google's Gemini and GPT-4o. But they're beginning to run up against technical roadblocks — in particular, computation-related roadblocks.

Why is this an interesting problem?



Language Models are Unsupervised Multitask Learners

Alec Radford^{1,2} Jeffrey Wu^{1,2} Rewon Child² David Luan² Dario Amodei^{1,2} Ilya Sutskever^{1,2}

OpenAI

Retentive Network: A Successor to Transformer for Large Language Models

Yutao Sun^{*1,2} Li Dong^{*1} Shaohan Huang¹ Shuming Ma¹

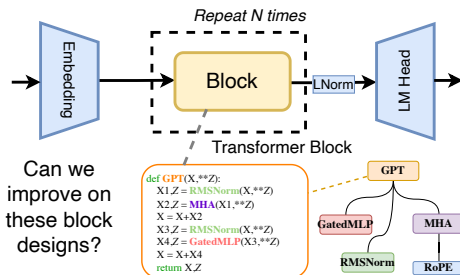
Yueqing Xia¹ Jilong Xue¹ Jiansong Wang¹ Furu Wei^{1,2}

¹ Microsoft Research ² Tsinghua University

<https://aka.ms/GeneralAI>

Why is this an interesting problem?

- Finding improved layer designs for autoregressive language models.



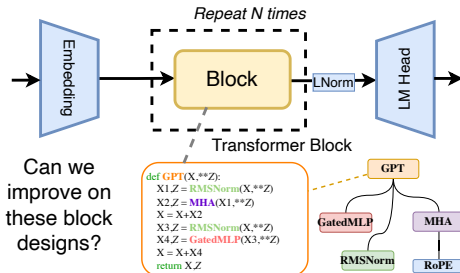
Ill-formed search space: huge unbounded design space.

Complex sampling process: literature understanding, coding skills.

Expensive verification: pre-training/evaluation, resource bound.

Language model architecture design discovery: how?

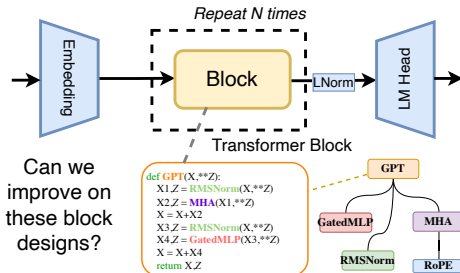
- Finding improved layer designs for auto-regressive language models.



Continuous learning loop: Generate new model ideas, implement them and verify through generative pre-training.

Language model architecture design discovery: how?

- Finding improved layer designs for auto-regressive language models.

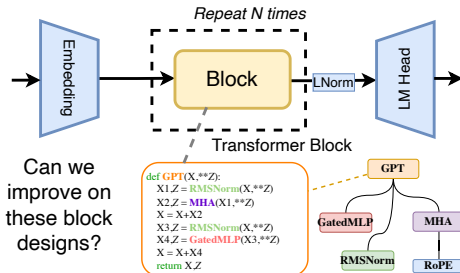


Continuous learning loop: Generate new model ideas, implement them and verify through generative pre-training.

- **Objective:** Find designs that improve on end-task performance.

Language model architecture design discovery: how?

- Finding improved layer designs for auto-regressive language models.

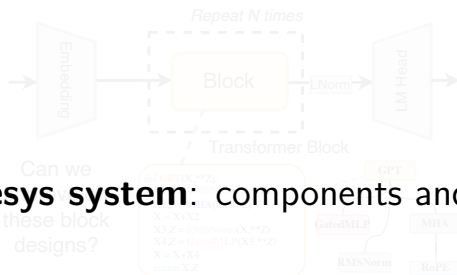


Continuous learning loop: Generate new model ideas, implement them and verify through generative pre-training.

- **Objective:** Find designs that improve on end-task performance.
- Start small, innovate then scale, **Ladder-of-scales** (LoS) approach.

Language model architecture design discovery: how?

- ▶ Finding improved layer designs for auto-regressive language models.



The **Genesys** system: components and principles

Can we
discover
these block
designs?



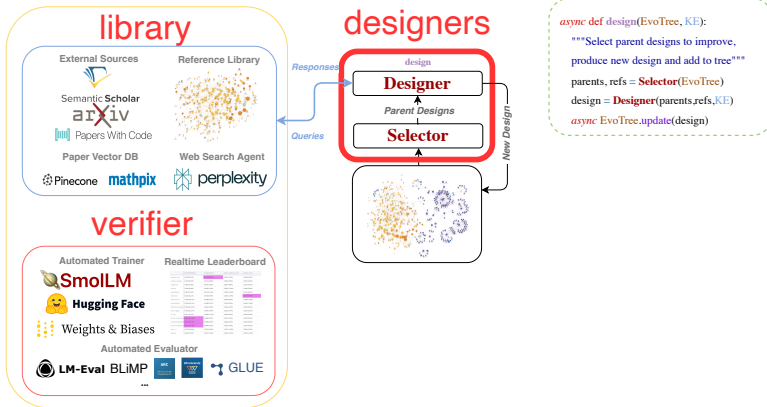
Continuous learning loop: Generate new model ideas, implement them and verify through generative pre-training.

- ▶ **Objective:** Find designs that improve on end-task performance.
- ▶ Start small, innovate then scale, **Ladder-of-scales (LoS)** approach.

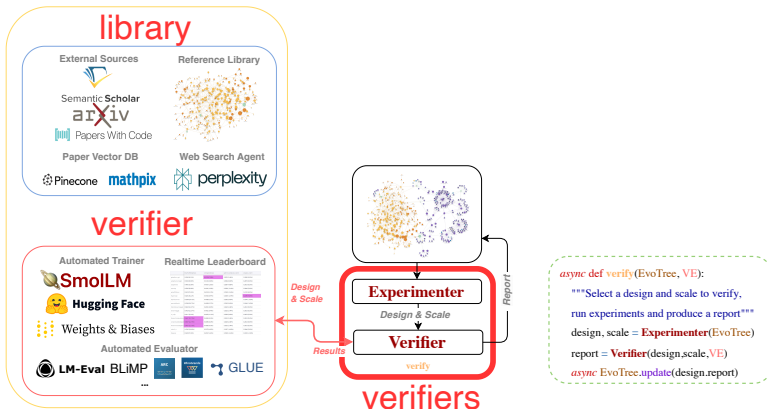
The Genesys system: core utilities



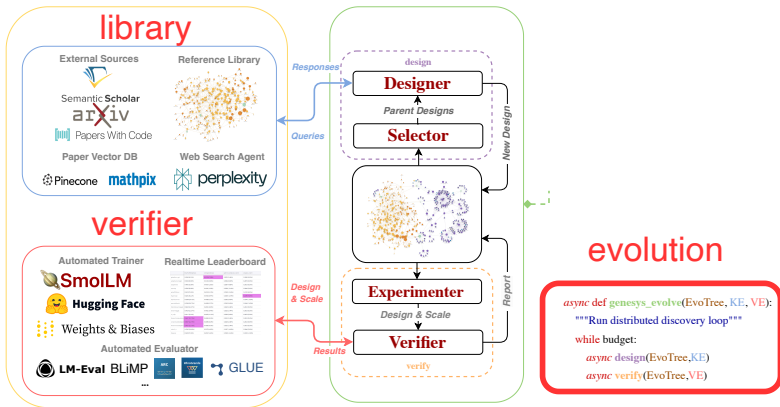
The Genesys system: agents



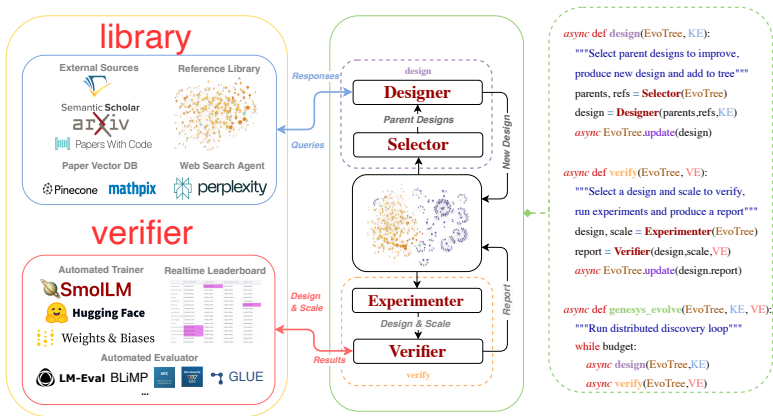
The Genesys system: agents



The Genesys system: distributed evolution



The Genesys system



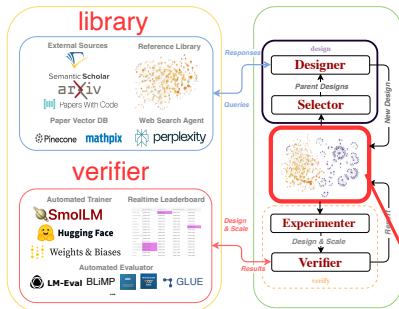
Experiments at a glance: 1,162 discovered designs (1,062 fully verified), 86K dialogues, 2.76M lines of code, 1B processed tokens.

The Genesys system



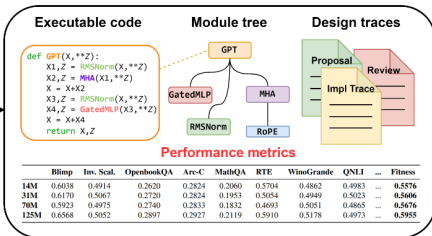
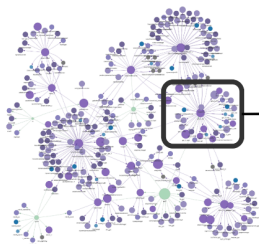
Experiments at a glance: 1,162 discovered designs (1,062 fully verified), 86K dialogues, 2.76M lines of code, 1B processed tokens.

Design tree: fully factorizable design space

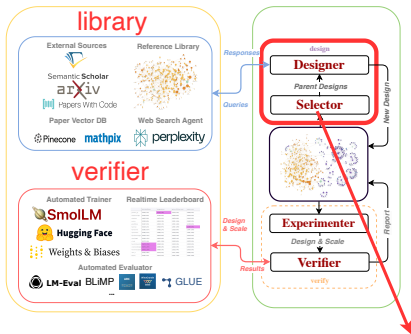


Code is fully factorizable,
representable as a unit tree

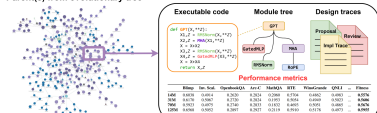
Fitness score: end task
performance



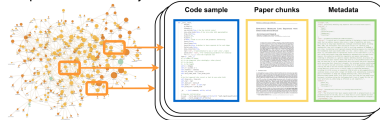
Designers: Proposer-reviewer architecture



Parent(s) from evolutionary tree



Examples from reference library



Proposal

1. VQW-GAT: Vector Quantized Hierarchical Generalized Autoregressive Unit Attention. The proposed VQW-GAT integrates explicit gating, vector quantization, and hierarchical memory architectures to enhance computational efficiency and scalability in autoregressive language models.

1.1. Motivation. Autoregressive language models (LLMs) have achieved remarkable success across various natural language processing (NLP) tasks. Considerable performance on Generalized Autoregressive Units (GATs) that process long sequences efficiently. However, as model sizes and sequence lengths continue to grow, existing GAT architectures face significant challenges in maintaining computational efficiency, memory scalability, and the ability to capture long-range dependencies. In addition, these limitations, there is a pressing need for an innovative GAT design that integrates explicit gating, vector quantization for memory compression, and hierarchical memory architectures. The proposed VQW-GAT aims to address these challenges by introducing memory retrieval, enhancing scalability while maintaining or improving current state-of-the-art performance metrics such as perplexity, accuracy, and inference.

1.2. Related Work. **1.2.1. SELECTIVE CONTEXT MIXING MECHANISM.** Selective gating has been effectively utilized to model the **Manley** [1] and **Flax** [11] to dynamically manage data representations based on input contexts. These techniques allow models to retain only relevant information, optimizing memory usage and computational resources. Integrating explicit gating units (GATs) can significantly improve their efficiency and focus on essential contextual cues, reducing unnecessary computations and model complexity.

1.2.2. VECTOR QUANTIZATION TECHNIQUES. Vector quantization (VQ) has emerged as a powerful tool for memory compression in large-scale models. Techniques such as **Principal Vector Quantization (PVQ)** [17] and **Channel-Based Vector Quantization (CBVQ)** [18] offer efficient methods for encoding high-dimensional data with minimal information loss. Implementing VQW-GAT enables memory representation, thereby reducing memory footprint and accelerating computation without compromising model performance.

1.2.3. Hierarchical Memory Architectures. Hierarchical memory structures, as demonstrated by **DeepSeek** [2] and **Neural Language of Thought Models (NeLoT)** [19], facilitate efficient handling of long-range dependencies by organizing memory into hierarchical blocks. Such architectures enable GATs to maintain fine-grained differentiation across different scales, reflecting their capacity to capture complex patterns inherent in language data while maintaining computational and memory efficiency.

1.2.4. COMPUTATIONAL EFFICIENCY AND SCALABILITY. While memory retrieval, vector quantization, and hierarchical memory can lead to significant improvements in computational efficiency and scalability. These models achieve faster inference times and reduced memory footprint, making them well-suited for deployment in large-scale language modeling. In adopting similar strategies, the proposed VQW-GAT aims to optimize computational resources and facilitate scalable model design.

1.3. Problem Analysis.

1.3.1. CHALLENGES IN EXISTING GATs

- Memory Inefficiency:** Existing GATs may suffer from high memory footprints, especially when dealing with long sequences, leading to high variability.
- Data Management:** Conventional data management in GATs may be inefficient for long relevant tokens, leading to unnecessary computations and potential loss of critical context.

Review

1.1. Comprehensive Analysis of the VQW-GAT Proposal

1.1.1. Innovative Proposition.

- The VQW-GAT uniquely combines selective gating mechanisms, vector quantization, and hierarchical memory architectures within Generalized Autoregressive Units (GATs). While each of these components has been successfully implemented individually in various models, their holistic integration appears to be novel. This combination approach addresses multiple facets of autoregressive language model limitations simultaneously, potentially leading to a paradigm shift in efficiency and scalability.

1.2. Problem Context.

- Selective Gating Mechanism: Models like **Manley**, **Flax**, and **Flax** and **General State Space Models (GSSMs)** have demonstrated the effectiveness of selective gating in managing long representation dimensions. The authors introduce novel ability to focus on relevant information, thereby improving memory utilization and capacity to handle long-range dependencies.
- Vector Quantization Techniques: Techniques such as **Principal Vector Quantization (PVQ)**, **Channel-Based Vector Quantization (CBVQ)**, and hierarchical methods in models like **Speech Transformer** and **CDSPPT** have proven successful in compressing memory without significant loss in performance. These techniques are pivotal for reducing memory footprint and accelerating computation.
- Hierarchical Memory Architectures: Architectures like **Segmented State Space Models (SSSMs)**, **NeLoT**, and **Neural Language of Thought Models (NeLoT)** effectively organize memory across multiple levels, enhancing the model's ability to capture complex patterns over extended sequences.

1.3. Theoretical Soundness.

- The proposal provides a well-motivated justification for each integrated component, asserting that the design decisions are grounded in established theoretical foundations. This includes detailed justification of selective gating, vector quantization, and hierarchical memory integrations, which collectively support the anticipated improvements in efficiency and scalability.

1.4. Alignment with Current Research Trends.

- The proposal aligns with ongoing research trends focused on enhancing model efficiency, scalability, and the ability to handle long-range dependencies. By addressing multiple challenges simultaneously, VQW-GAT positions itself as a frontier advancement in autoregressive language modeling.

1.5.1. CONCLUSIONS

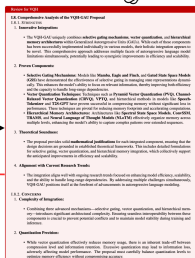
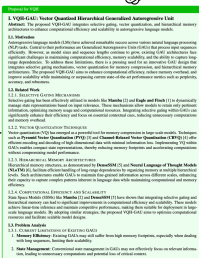
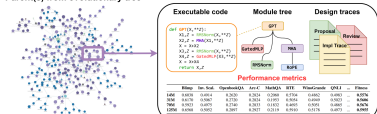
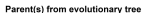
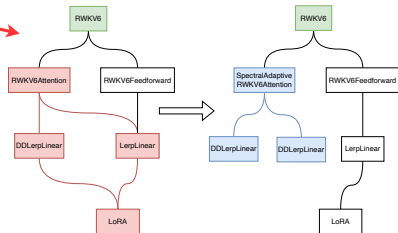
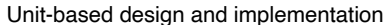
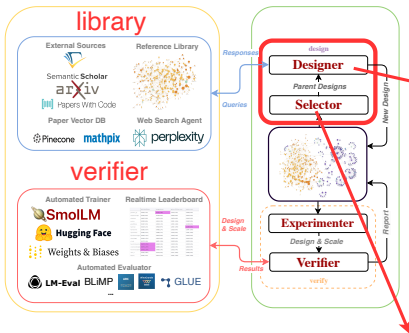
1.5.2. Complexity of Implementation:

- Combining three advanced mechanisms—selective gating, vector quantization, and hierarchical memory—may introduce significant architectural complexity. Ensuring consistent interoperability between these components is crucial to prevent potential conflicts and to maximize model stability during training and inference.

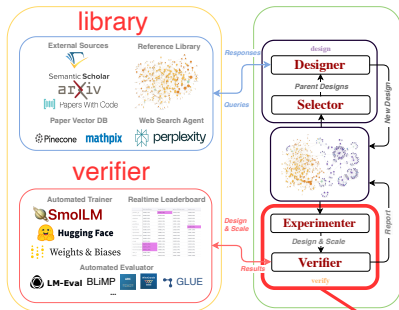
1.5.3. Quantitative Provisions:

- While qualitative efficiency reduces memory usage, there is an inherent trade-off between compression and loss of information retention. Theoretical quantification may lead to information loss, ultimately affecting model performance. The proposal must establish baseline quantitative benchmarks to measure memory efficiency without compromising accuracy.

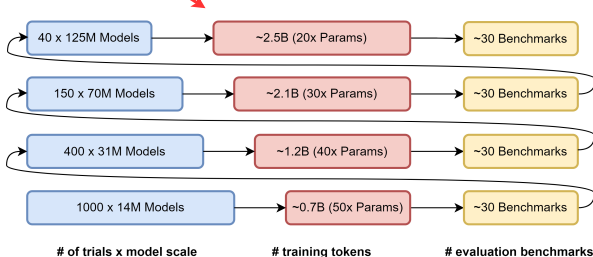
Designers: Planner, coder, observer



Verifiers: budget sensitive scaling



Ladder of scales (LoS)
approach



A sketch of the results: **end task performance**

Have we made any discoveries yet?

A sketch of the results: end task performance

	Blimp	Wnli	RTE	WG	CoLA	SST2	WSC	IS	Mrpc	avg.
<i>Random</i>	69.75	43.66	52.71	48.78	50.00	49.08	49.82	50.03	31.62	49.49
GPT	92.70	60.56	62.80	52.17	53.24	54.13	56.76	55.31	68.38	61.78
Mamba2	83.22	63.38	63.88	51.22	55.94	56.58	57.12	53.85	67.89	61.45
RWKV7	88.76	61.97	60.21	49.80	54.25	55.32	54.57	57.00	68.38	61.14
RetNet	85.16	61.97	61.35	50.51	56.29	55.43	56.03	54.95	56.37	59.78
TTT	86.13	63.38	55.23	50.75	55.55	56.35	54.93	55.31	59.80	59.71
VQH	94.37	59.15	59.91	50.28	54.25	53.56	53.83	49.45	56.62	59.05
HMamba	83.74	64.79	61.35	53.59	54.69	57.04	56.40	54.58	59.31	60.61
Geogate	90.95	59.15	61.35	52.72	54.25	55.32	58.96	54.95	68.63	61.81
Hippovq	87.96	50.70	59.91	50.28	54.25	55.73	53.83	55.68	69.88	59.80
SRN	80.83	65.52	59.55	50.75	54.45	52.98	56.03	54.95	61.03	59.57

Table 3: Performance of human designs and discovered models on various Benchmarks (350M Parameters, 50B Tokens). Metrics indicate accuracy percentages. Bold and underlined denotes the top and second best, italics denoting worst.

new designs

A sketch of the results: end task performance

	Blimp	Wnli	RTE	WG	CoLA	SST2	WSC	IS	Mrpc	avg.
<i>Random</i>	69.75	43.66	52.71	48.78	50.00	49.08	49.82	50.03	31.62	49.49
GPT	92.70	60.56	62.80	52.17	53.24	54.13	56.76	55.31	68.38	61.78
Mamba2	83.22	63.38	63.88	51.22	55.94	56.58	57.12	53.85	67.89	61.45
RWKV7	88.76	61.97	60.21	49.80	54.25	55.32	54.57	57.00	68.38	61.14
RetNet	85.16	61.97	61.35	50.51	56.29	55.43	56.03	54.95	56.37	59.78
TTT	86.13	63.38	55.23	50.75	55.55	56.35	54.93	55.31	59.80	59.71

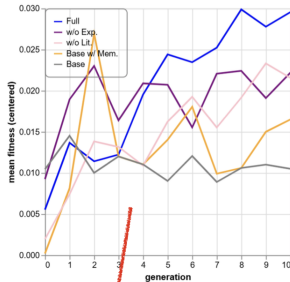
Result: Yields designs competitive with human ones

VOH	94.37	59.15	59.91	50.28	54.25	55.36	53.83	59.45	56.62	59.05
Geogate	90.95	59.15	61.35	52.72	54.25	55.32	58.96	54.95	68.63	61.81
Hippovq	87.96	50.70	59.91	50.28	54.25	55.73	53.83	55.68	69.88	59.80
SRN	80.83	65.52	59.55	50.75	54.45	52.98	56.03	54.95	61.03	59.57

Table 3: Performance of human designs and discovered models on various Benchmarks (350M Parameters, 50B Tokens). Metrics indicate accuracy percentages. Bold and underlined denotes the top and second best, italics denoting worst.

new designs

A sketch of the results: system and design analysis



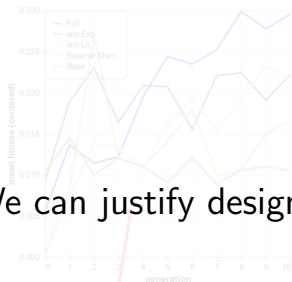
system stability

	Valid	Attempts	Costs	LFC
Full	92%	2.6 (± 1.1)	15.0 (± 18.5)	181 (± 44)
No FF	73%	3.0 (± 1.7)	7.9 (± 7.1)	75 (± 29)
No Pl.	91%	2.6 (± 1.1)	16.0 (± 20.8)	218 (± 69)
No Ob.	89%	2.6 (± 1.1)	12.1 (± 20.1)	211 (± 67)
No SC	30%	2.4 (± 1.0)	2.9 (± 4.7)	167 (± 33)
Simple	6%	1.1 (± 0.2)	0.3 (± 0.3)	49 (± 15)
Library	-	-	-	220 (± 136)

Table 3. Agent benchmark results. Bold and underlined denotes the top and second best. "Library" stands for our reference library with 180 designs providing core block code.

successful code
generation rates

A sketch of the results: system and design analysis



We can justify design, empirically and formally.

system stability

	Valid	Attempts	Costs	LFC
Full	92%	2.6 (± 1.1)	15.0 (± 18.5)	181 (± 44)
No FF	73%	3.0 (± 1.7)	7.9 (± 7.1)	75 (± 29)
No Pl.	91%	2.6 (± 1.1)	16.0 (± 20.8)	218 (± 69)
No Ob.	89%	2.6 (± 1.1)	12.1 (± 20.1)	211 (± 67)
No SC	30%	2.4 (± 1.0)	2.9 (± 4.7)	167 (± 33)
Simple	6%	1.1 (± 0.2)	0.3 (± 0.3)	49 (± 15)

Library 220 (± 136)

Table 3. Agent benchmark results. Bold and underlined denotes the top and second best. "Library" stands for our reference library with 180 designs providing core block code.

successful code
generation rates

Please come to the poster to learn more

Thank you.

References I

- Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Pham, H., Dong, X., Luong, T., Hsieh, C.-J., Lu, Y., et al. (2023). Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233.
- Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J., and Ha, D. (2024). The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*.