On The Way To The General Artificial Intelligence: Autonomous Deep Learning Apps

1st Gabriel Alves Castro - 170033813
departamento de Ciência da Computação
Universidade de Brasília
Brasília, Brasil
gabriel_alvesc1@hotmail.com

***Abstract*—The use of deep learning as an app generator is explorated in this work, being used several deep models and compared one to another. The hipothesys is that if we can utilize deep learning to generate app's in an autonomous way, so we can mantain the research of deep learning next of the market and at the same time we will find more general deep architectures like this task demands, helping us to achieve the general intelligence. Several jobs of deep rendering are comparated and used like inspiration for the construction of autonomous deep learning apps (ADLA).**

***Index Terms*—Deep learning, convolutional neural networks, image-to-image translation, Ui generator, deep rendering, autonomous deep learning apps, conditional adversarial networks**

I. Introduction

Several works were made on the field of graphical user interface (GUI), game engines and so on with deep learning. One of the more famous works on deep learning rendering was made by (Ming-Yu et al.,2017) [3], where they did an deep learning to solve the task of by some image generate other image, like the example of any image-to-cat image. (Tianming et al, 2021) [4] used a databank of UI (user interface) interfaces images to generate new original UI interfaces from some images, given a kind of smartphone app. (Jieshan et al, 2020) [5] trained a deep learning network to recognize GUI elements and give it labels, to improve the accessibility of user interfaces.

(Hiroharu et al., 2020) [8] cite the neural rendering, with images originated from data proposed by (SM Ali et al.,2018) [9] like a new vibrant field of research with several potential applications. The work show a application of neural networks to represent scenes and generate new scenes from these scenes. What is a few semeliant to our ADLA netowork. (Seung Wook et al., 2020) [10] was the most similar work with ADLA finded, that created an Generative Adversarial network to simulate games, our approach pretend to expand this concept for any kind of application with mockups, while this job made the learning process based on observation of some game and it imitation. At the same time, any game can be modeled like the necessary data to ADLA generate it.

my thanks to the University of Brasília and to the entire Brazilian scientific community

With these evidences we can see that neural networks can have the capacity of construct entire applications with the correct data and architecture. With this in mind we propose a new front of application: The use of neural networks like the responsibles for the engines and image generation of apps, the networks on these field can be named like autonomous deep learning apps (ADLA). Therefore we can describe the taks like, given a set of apps inputs, images and constraints (a mockup) we have to generate a network to construct the own app. Formaly, we will have the function: f(I,Im,C) = Live_app, where I are the inputs of some planed app, Im are the interfaces of the app and C are the possibles constraints. It is possible, like on our ADLA proposition that the constraints be inserted on the own images data.

We see these kind of research like an oportunnity of improve and inovate on the kinds of neural networks, include arriving on more general deep learning models, more focused on more general tasks, more efficient neural networks and more attachable networks. Furthermore, apps have a big market, so we can bring the field of deep learning even closer to the resources coming from this market. Further, a ADLA network can reduce the time for construct apps, because if we have one ADLA model, so we can fastly generate several apps by just generate new mockup (a necessary step in app development) and tuning the model. Finally, given the capacity of deep learning to find hard patterns, we can even become capable of construct totally new and innovative apps, like examples on (Seung Wook et al., 2020) [10].

II. Metodology

For this work was created a small databank of images representing a small application. A dataframe was created to represent transitions by clicks, where given a click and a application screen the next application screen was given. The objective of the work is train a deep neural network to acts like an application, so the task can be formalized like: f(C,Sc) =

Sn, where C is the click position, Sc is the current screen and Sn is the next screen of the application. Consider each screen like an rgb matrix. On this work, the Sc was represented with histograms, to make the data input small, and it brings new results, but it is not necessary. Other possibility is to make the Sc an latent representation of the image, by a autoencoder.

Several models and his limitations were tested using the keras package for python [1] on a personal machine with 16gb ram and a intel core i7 8° generation, first the most simple and obvious models of deep learning were applied: dense neural network, autoencoder of dense network, convolutional neural network fallowed by a dense neural network where the click input was concatenated. The results and conclusions of this phase fallows on the next sessions.

Because of the length of the models with more than one million parameters, the RGB values of the screen image input were divided by the max RGB image value (it can be made because RGB values are ever positive values), to don't saturate the neurons of the network.

For each simple model, the stochastic gradient descent and the adam algorithms of otimization were tested, and for loss functions the mean squared error was firstly utilized. The dropout method too was tested to try augment the accuracy.

Finally, Generative Adversarial Networks (GAN) were tested in a job inspirated by (Ming-Yu et al.,2017) [3] that used conditional GAN's, in our case the new screens have to be generated by conditioning the generation to the click position and current screen. First we used just binary cross entropy like loss function, but after for the generator was added an mean squared error component for make the generator ever approximate to the true image, what brings better results. Aditionally, L1 regularizers was used on generator and discriminator. The discrimiantor was tested without L1 regularizers but it doesn't gave good results. Batch normalization too was added betwenn each layer of the generator. So we can write the loss function of the generator like:

$$LossG = binary\_crossentroy(1, D)$$

$$+$$

$$mean\_squared\_error(truth\_images, generated\_images)$$

Where 1 indicates that the generator tries to maximize the chance of the discriminator D predict the generated image like a real image.

A. Proposed Architecture

Inpirated on Conditional Generative Adversarial Networks (CGAN) we propose an new architecture based on our kind of data. the objective is to generate frames with just an screen and a click coordinate, by neural networks. We can think that other approachs can be better for the obective of put all the logic of an application inside a network, (Tianming et al, 2021) [4] demonstrate that is possible to use structural data to obtain better results, or we can even just separate the networks on several subtasks and tries to fit to the expected result. On this case we can consider that the model does not are generative, because we don't use noise on his construction.

But (Jorge et al., 2019) [6] and several other works related show us that deep learning can be Turing complete, what mean's that with clicks and images we have the enough amount of data to do an enormous amount of applications. From this we decided to make a network with just the mentioned inputs and output like a first work, and we propose to advance on the complexity of the model and field on the future.

We divide the network on three pieces: The discriminator, the generator and the extensor. The discriminator is a feed-foward network, with one image input Im1, that are the image to the discriminator decide if is true or false. The input is applied to some layers of convolution until arrive on a dense layer of just one neuron, to classify if the generated image is false or truth. So the function of the discriminator can be described like f(im) = boolean. On the convolutional layers was added a regularizer L1 with 0.02 alpha, and to work in a balanced way the discriminator have to have the same amount of convolutional layers to the generator, with at least a similar number of filters to that used in the generator. We found it empirically.

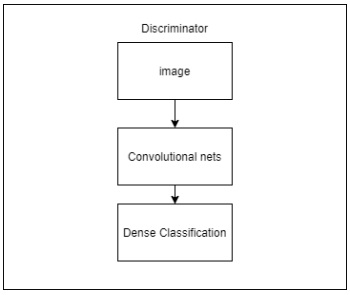

Fig. 1. Discriminator Architecture

The extensor actually make part of the generator. The extensor is an dense network that has as input the coordinates positions of some click connected to a dense network, with layers of at least one hundred neurons. The objective of this network is to augment the capacity of the clicks on make modifications on the final result, balancing the relative number of neurons influenciated by the click data (just two numeric inputs) and the image data (255 numerics inputs). We see too, that this network can speed up the convergence of the entire ADLA network.

Finally, the generator is a network with two inputs: The output of the extensor and the current screen that was reduced and transformed into a vector histogram. These two inputs are combinated and applied to a dense network, and then resized to an image shape and passed to an inverse convolutional network to generate the output image, that is the next screen of the application. So the function of the generator can be described like f(click,im') = im".

The two networks were trained with the binary_crossentropy loss on the more basic application. Was tested the efficiency

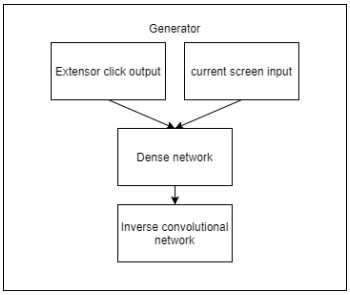

Fig. 2. Generator Architecture

with L1 regularizers like on (Ming-Yu et al.,2017) [3], and without the regularizers. But do not was implemented a discriminator by pieces of a image, but for a entire image. The adam optimizer was utilized from the fact that it is more suitable to fast convergence (Diederik P. et al., 2014) [7]. The binary_crossentropy was not enough to convergence, so was added an mean squared error to the generator loss, like talked before.

All the models were constructed using a ReLu activation, Leaky ReLu activation or sigmoid Activation, by times with a combination between them for test. But Leaky ReLu brings better results, like expected on the literature of GAN's.

To train the network were created from the datas of images and button positions and lengths, datas of clicks, screens and next screens. The data was shuffled, and contain from 35 to 800 rows, it depends of the approach tested. The options for the length of the networks were limitated, because if much big the memory of our personal machine were exploded

B. images of application

The data tested int this job were from the app created by us "What yours feelings tells us about the future general artificial intelligence". It was designed to be a simple application, with 16 screens, where the user can choose some options on the initial page, and given the options chosed the user will receive back a image and a text. A example of the initial page without any button clicked and one next screen is given bellow.

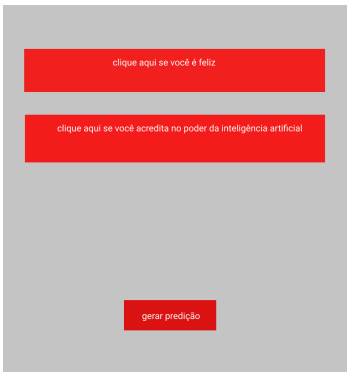

Fig. 3. Initial screen of the app - A image data for the model ADLA learn

These screens were constructeds like a mockup with the figma application [2] that is a famous application to build

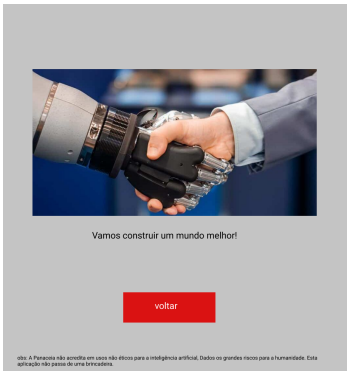

Fig. 4. Screen with some of the previsions - A image data for the model ADLA learn

mockups for apps and sites. Look that the objective is to make the generator understand that by click on some button, some new image have to be generated, like make the button green to indicate that it was clicked or see that a given button was clicked and by this generate the rigth image prediction after click on "gerar predição" button, that will be something like the figure 4.

III. Composed errors

The process of execute an application with ADLA fallows like: An encapsulament of ADLA network generator on a while Loop, a printer for the images, and a input for click mouse coordinates. the initial screen is given and printed. At each step on while loop, the actual screen is reduced to a vector histogram, and a click is waited. When a given click happens the coordinates of the click and the vector histogram are passed as input to the generator, that generate the next screen. Again the procees repeat, with the new rendered screen being used to generate the next screen. Where the first screen of the process is the initial page screen image without being generated by the generator.

We can see that with this process, we can expect acumulated errors, and will have a minimum level of quality for the images generated to the process to work. Because that on the current state of the work we cannot generate good enough images we couldn't test this process, and it will be a future work.

IV. Generator plus RNN process

An alternative that we begin to test is to construct an RNN network that will remember the current screen and will generate the next screen histogram. It network have like inputs the click and the screen histogram vector, and like output the histogram or a latent representation of the next image, or the own next image. After that a generator will takes the histogram of the next screen or a latent representatiton and generate the entiry screen, if it was not generated by the RNN network. On (Seung Wook et al., 2020) [10] one of the components of the GameGun netwok are similar to that, but here we have a much more simple network. It will not be showed on this work.

The advantage of this network is that we can use on it the robust theory of Lyapunov to improve the results, and we can divide the steps and reduce the amount of data, what can be better. With a small change on this network we can generate an autonomous agent that take the action of itself (click) or the user action, and generate an image and an action (click), or even just an new image that represent his action. It network can be composed with the network created on this work, and produce more complex apps with several possibilities that we will talk on future works.

V. Results

For the most simple models, the results don't were so good. All the time the functin fell on a local minimum, bring a loss for the mean squared error greater than 2000. It means that all the later screens were being combinated on just one image. A example of this problem is given on the image bellow, generated by these first models:

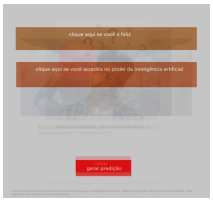

Fig. 5. mixture of screens: The simple models generate every times the same image

This result wields us the though that for this task is necessary some kind of memory, and a better though about the loss functions. At the same time, like each prediction can be from very distinct images to other very distinct images, maybe more complex models or even separated models can be used, from the fact that this task can be impossible for feed-foward models on a personal machine. At this point, we can see that this pressure can create the necessity of more general deep neural architectures, that can solve these kind of problems. These supositions have like evidence the work of (Seung Wook et al., 2020) [10] that use three components, with one of the components beig responsible just for memory, and it worked well. Furthermore, recurrent networks can plays important tasks, from the fact that the big majority of the applications have logical transitions and animations and need memory and dinamical execution to achieve the necessary results.

A. CNN with histograms of images

Even with the before results, we continues to try with simple models, and we get better results. After add an histogram vector like image rather than the image and L1 regularizer on the generator, we was abble to train it with just an direct process using mean squared error. It gave us a loss of 300 what is much better than the previous results. But it gave us blurred images, like expected by this kind of loss function, the legends and texts buttons were not visible on the generated images.

B. ADLA network results

The training process takes more than three thousand steps, taking more than one day to arrive on best results, but it can be even better, because when we stoped the training the network was still doing improvements on the images (at every 100 steps passed the improvement is visible). This time is because we just can use a cpu to do the trainement, on a correct device it can be so far better.

After the training we obtained better results, with more visible legends. But the images yet are generated with a mixture of images on some cases, and are not perfect. We imagine that is possible to improve this result with changes on the structure of the network, or just giving more time to train the networks.

Other possible options are expand the layers on depth and length with a better machine, or create parallel layers with spatial softmax on the final to create distincts paths to the network create, for example, the distincts possible images and decide what is the better option on a given moment, what was made on (Seung Wook et al., 2020) [10] work. But because our job are a more simple proposition based on PatchGAN (Ming-Yu et al.,2017) [3] work we let it for an further work.

From the fact that our application have just one possible transition of screen with some distinctions for each kind of screen, we does not finded necessary to apply an Recurrent Neural Network or some variant. But a natural expansion for the ADLA model proposed is the addition of a Long Short Term Memory (LSTM) layer on the generator component, we have it on the GameGan (Seung Wook et al., 2020) [10] model, and it can supply the ADLA model the capacity to generate animated apps. Like talked before.

After solve the problems of convergence a next natural step of this work is test the network on more mockup's of other apps, and see what will be the results. Furthermore we can test what will be the acumulated errors, and lidate with them.

C. Images generated ADLA CNN X ADLA GAN generator

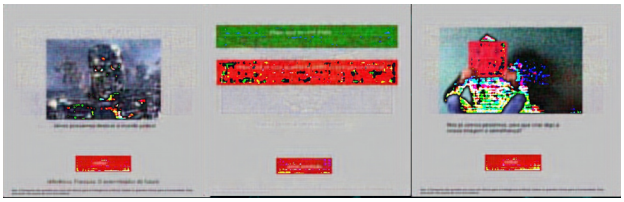

Fig. 6. Generated images from the ADLA CNN with blurred legends

Fig. 7. Generated images from the ADLA GAN with better legends

We can see that the images generated are not perfect, but the logic of the buttons, and the images itself with legends and texts are captured by the networks, what are a very good result (it means invisible programations, just from the mockups). It means that is possible to a person just plan the mockups of some app, and without knowing how to program, create a program with ADLA network.

Then GAN method seems to be the better path, just need more time to convergence or some changes to brings the expected results, but ir have the capacity to solve the blurry problem. We hope show better results on future works, but the objective of this work is completed: We prove that neural networks can learn logics of an application, and generate the application itself, what is needed now is just improve the qualiteness of the app generated, and make it more complex.

D. Future works

We can see enormous oportunities of reasearch on ADLA field, based on Guigan network (Tianming et al, 2021) [4] and gameGAN network(Seung Wook et al., 2020) [10] we propose that is possible to train ADLA models on serveral domains of apps, and use transfer learning to create entire new apps based on mockups with beatiful UI designs and animations, even without a good mockup.

The inclusion of LSTM on the model can bring us the possibility of create better animated app's and have to be explorated on the ADLA field on further works. In conjunction with that several ways of transform the images on input or build the networks have to be explorated to improve the efficiency of the models, because apps have to be small applications or be served like a service.

Other perception is that much apps need of several data inside it to work, like text data. And need to remember important things or even do communications one another, and receive several kinds of inputs. From these facts, the ADLA model proposed have to be expanded to lidate with text data, input options and communications (from output to input via socket). For do it we see three paths: Combinate the models with traditional programation and mixture kinds of input and output, create several distinct and separated models to acomplish distinct tasks, or create more general networks, even that don't use gradient strategys or the traditional deep learning. We believe that big advances on deep learning are on the third approach, and we think that it is the better path because of the flexibility power of conexionist intelligences. Nonetheless, the second and first approach is not bad.

VI. Conclusion

On the start of the job we had the hope that the most simple deep learning networks can do the work of the most simple ADLA application. And it was confirmated on this work. It don't mean that we have to stop in it, we need to try more. The more complex models, like conditional adversarial networks seems to be promising, and have to receive a focus and be expanded on the future. The promise results, and promised future works can bring us to a new stage on deep learning era, to more general artificial intelligences. The ADLA apps can be at the same time the field of research for deep learning and the source of resources for the researchs.

References

[1] https://keras.io/.

[2] https://www.figma.com/

[3] LIU, Ming-Yu; BREUEL, Thomas; KAUTZ, Jan. Unsupervised image-to-image translation networks. In: Advances in neural information processing systems. 2017. p. 700-708.

[4] ZHAO, Tianming et al. GUIGAN: Learning to Generate GUI Designs Using Generative Adversarial Networks. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, 2021. p. 748-760.

[5] CHEN, Jieshan et al. Unblind your apps: Predicting natural-language labels for mobile gui components by deep learning. In: 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE). IEEE, 2020. p. 322-334.

[6] PÉREZ, Jorge; MARINKOVIĆ, Javier; BARCELÓ, Pablo. On the turing completeness of modern neural network architectures. arXiv preprint arXiv:1901.03429, 2019.

[7] KINGMA, Diederik P.; BA, Jimmy. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[8] KATO, Hiroharu et al. Differentiable rendering: A survey. arXiv preprint arXiv:2006.12057, 2020.

[9] ESLAMI, SM Ali et al. Neural scene representation and rendering. Science, v. 360, n. 6394, p. 1204-1210, 2018.

[10] KIM, Seung Wook et al. Learning to simulate dynamic environments with gamegan. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. p. 1231-1240.