EPISODE 1175

[INTRODUCTION]

[00:00:00] JM: For several years we have had the ability to create artificially generated text articles. More recently, audio and video and synthesis have been feasible for artificial intelligence to generate. Rosebud is a company that creates animated virtual characters that can speak. Users can generate real or fictional presenters easily with Rosebud.

Dzmitry Pletnikau is an engineer with Rosebud and he joins the show to talk about the technology and engineering behind Rosebud.

To support Software Engineering Daily with a paid subscription, you can go to softwaredaily.com and become a paid subscriber. You can get ad free episodes, and it would really help us out. Thanks for listening.

[INTERVIEW]

[00:00:48] JM: Dmitri, welcome to the show.

[00:00:49] DP: Thank you for having me, Jeff.

[00:00:52] JM: You work on Rosebud. Rosebud is a system for generating the faces of human models. Why is that useful?

[00:00:59] DP: That's a good question. So what Rosebud AI does extends beyond that, beyond just generating human faces. Our vision is to enable people to express their creativity at the speed of thought, so to say. And generating human faces is obviously a part of it, because like no matter what kind of art you generate, there are likely faces and characters in it. If it's a video, video recording or like a sketch for a movie or a book, like an illustrated narrative of any kind, or an ad, or some kind of poster, you have to have a person in most cases. Have to have a face. So that's why the first product that – Well, one of the first products that Rosebud AI has been focusing on has been generating human faces and allowing users to edit how those faces look, but would have launched recently goes beyond that, specifically the talking heads product is mostly about videos and driving the artificial or real characters, human faces with your own voice or with your own video or with just the text. But you right that it all starts with the face.

[00:02:16] JM: Tell me more of the types of customers, the types of users that would need these AI-generated models.

[00:02:22] DP: There is of course the fashion use case. We have been focusing on that for some time, and now we are looking at other audiences. There is a lot of interest. We have customers from different niches and people coming up with use cases. We have people who are playing games and they want to animate, essentially have a character. They want to animate with kind of an alternate persona. That's one use case.

So what I want to say is we build the technology, which can drive a great variety of different use cases. It's very flexible. And the specific markers that we address are buried. I’m probably not the best person in the company to answer this question. Lisha is the founder of Rosebud AI. She's much closer to the customer, customers, and talking to customers every day. Should be in a better position to answer this question.

[00:03:27] JM: That’s okay. I think we've got the product use case and we can start to dive into the engineering a little bit more. I think it's worth going over the topic of generative models. Could you explain what a generative model is?

[00:03:42] DP: Generative model is a machine learning model which can generate video or audio or visual still frame unconditionally essentially from noise or conditionally, conditionally in a sense that we can control what it outputs. As an example of very popular generating models began at StyleGAN, and strictly speaking you can run them from any noise vector, uses input and get a realistic image out of it. But the most powerful use case for these models is when you have a way to control what's been generated. So instead of getting some random image of human, you can choose to generate a human who is smiling or who have short hair, a long hair. Does give you a good impression of what a generating model is.

[00:04:36] JM: Absolutely. Tell me about some of the applications of generative models that have arisen over the last couple years when they become popularized.

[00:04:45] DP: We have seen a lot. So I think one of the first products in the space was – First applications in the space was just generating fake people and using them as avatars, which can be used for users who want to maintain a certain level of uh anonymity or pretend to be someone else. That's more of a kind of negative use case, and yeah, just generating art. There's a well-known project called Art Reader. In Art Reader, people just try to export the space of generated model and see what interesting images they can find there, cool faces or cool abstract compositions or landscapes.

At Rosebud AI, we use it as a building block, as a component to generate rich visuals. So it's part of what we do. It's not like the whole thing. We have a lot of logic and machine learning going on beyond just generated models. It is an important part of pretty much every product we launch.

[00:05:50] JM: So Rosebud has started as this generative model system for creating human models. Can you walk me through the engineering stack required to create a human model?

[00:06:04] DP: Are you interested in full – Well, when we're talking about human model, we can talk about just the face or a full body. And the face is a strictly simpler, easier problem to handle because –

[00:06:15] JM: We can focus on face.

[00:06:17] DP: So with the face, the reason why I say face is easier and simpler to generate is because face is a solid body in a sense. Like geometrically speaking, it's symmetric. It's easy to align and orient it, which makes it an easier target for machine learning algorithms. So to generate a face, say, with a StyleGAN, the project is open source. StyleGAN was developed by NVIDIA, and they have iterated quite a bit on the project. So it's probably the most popular way of generating faces, like completely artificial faces. You check out a GitHub project. You can do it even in the Google collab and get the weights. The ways for the model are also publicly available, the ones that NVIDIA has prepared. And you can start generating.

The process itself is a straightforward. You sample pseudo random vectors and you push them through the model, and the model gives back the pixels. And that's how you get the face. If you're talking about controlling this output, then the space – The model has a completely random, sort of random Gaussian vectors as inputs. But then there is an intermediate step of model generation. This intermediate step is called D-latents. D-latents have interesting properties, which NVIDIA team has discovered, and they can be controlled and combined. So essentially if you have a D-latent kind of intermediate representation of the face, which is a 512 dimensional vector. You can do some algebra on them so you can have two faces represented by two 512 dimensional vectors. You can interpolate between the two and generate the frames, which gradually change from looking like one face to looking like another face. And if you have three vectors, you can try to explore the space in between and see how you can combine the two appearances to get some completely new appearance. So that's where the question of we are moving from the space of like trivial generation to the question of, “Okay, how can we control this exploration and how we can expose that as a UI to the end user if the end user is interested in generating every particular, very specific phase. How do we let a non-technical creator, artist explore this space efficiently and intuitively?” That's something that we have been working on, and that's visible in several of our products.

So specifically if you're talking about creative app, which is centered on editing human faces, we have the features such as moving along the directions of facial expressions. So that is very similar to what I was talking about. In our model we have a lighting space. We can let the user explore the space along predetermined directions. And starting with one face, explore how this face may look like if it has a wider smile or more of a sour face. But it is a very interesting and I would say open problem currently in machine learning how to efficiently explore the latent space of generated models. Because generated models are expensive to train from scratch. The datasets involved are huge. And if you're talking about multi-medium – Well, like visual, audio or video generation, datasets are big and the compute requirements to train the models from scratch are high. So it is virtually impossible to train a model from scratch for very specific use case. The question becomes how can we train one model? Let's say it’s very good at generating faces. And how we can then control it to get to the very specific use case to a very specific product that is useful for the people?

[00:10:16] JM: What have been some of the difficult engineering problems in building out the ability to generate these faces?

[00:10:25] DP: Generating faces themselves is quite straightforward. I would say the difficulty comes in what we do with the faces as the next step. So let's say if you have a face – And let's say you start with an image and you want to edit a face and an existing image. To do it efficiently with a generative model, like one way to do it is to find a latent representation of this face in the latent space of this particular generator model or like the whole body and then doing some algebra in the latent space to make the edits which you want to achieve. Finding the latent corresponding to the pixels is a pretty hard task, because the space of pixels and the space of latents in the generator model are match through like highly non-linear generator module.

And in general case, finding a latent or representation of given pixels is computationally intensive. It's an optimization problem in itself, and it is similar to – In its compute requirements, it's similar to training or optimizing the model. Essentially we are traveling in the multi-dimensional latent space. Trying to find the local optimum, and that's computationally intensive. So that's something that we also experienced in our face editing products. There's no really one great solution to it. It goes back to the user experience and how to frame the product in such a way that the process that can take minutes is acceptable to the user. So it comes down to designing the product from the ground up for the user to expect that and in the UI and how the product works, account for that, make it a great experience for the user in the end. So that's one thing.

Another thing is once – Let's say you generated the face, what you want to do with this face? Like if it's just a headshot and you needed a headshot, then you're done. That's great. So let's say if you want to use a fake face as an avatar, you're done here. But in most cases, if you generate a face and you control this generation, you want to do something with it. In particular, put it back into a video frame or a still photo, and that takes – That can be computationally intensive as well. And usually it's hard to transfer the GPU. So the question becomes of, "Okay, if we have the technology, let's say we have machine learning endpoint that generates faces and we have another machine, another endpoint, not necessarily even machine learning, that puts this phase back uh into the image. How do we efficiently connect this endpoints? Because at this point we transfer high resolution images between the two." So the challenge becomes collocating them in the network and figuring out a way of maybe using some shared storage instead of sending the images over the network back and forth between the endpoints.

And the reason why we even have this challenge in the first place is that at Rosebud AI we are users of microservices architecture. So all our machine learning endpoints, microservices that run in the Kubernetes cluster. And the processes that we automate, span, always span multiple services, multiple endpoints.

[00:13:55] JM: Can you tell me more about your choice of tooling across your stack, particularly the machine learning tooling?

[00:14:04] DP: Sure. It is a good question, first of all, because what the technologies that we use often dictated by the availability of tooling. So we try to be very pragmatic in our choices of technology. And in machine learning we're kind of locked into the Python ecosystem, of course, because that's what modern machine learning research uses.

As much as everyone else, we of course use notebooks, Jupyter Notebooks, which run in our Kubernetes cluster on the cloud or in the cloud. And beyond that package, our endpoint in Flask. So Flask serves as an HTTP adapter essentially, controlling the entrance into the entry point into the machine learning logic itself. And then everything is orchestrated by Kubernetes. I guess the tools that we use on a daily basis are mostly DevOps tools, which facilitate deploying development versions of the endpoints. And even during the development, because as developers, we don't necessarily have a GPU locally. So I’m using Macbook Pro as my main development environment. And I don't have a GPU. Aand some models must be run on the GPU. So in that case, we fall back to essentially cloud native development where we use scaffold with Kubernetes and hot reloading of the code, where if I edit the code, it's uploaded to the cloud, to the Kubernetes cluster. And the running image, Docker image, is patched with my changes and I can immediately see – Pretty much immediately, like within seconds, I can see the updated endpoint that I’m working on running in the cloud.

I would say as far as tooling goes for machine learning specifically, we try to do as much research as possible or as much experimentation as possible in the Jupyter Notebook, because that's the easiest, most straightforward environment for experimentation with machine learning models. And then once we are like very close to productionizing our work, we either do local flasks-based endpoints or cloud native development. And the reason why Jupyter is a much simpler environment for this kind of experiments is that when we talk about machine learning logic and models, there is a significant latency of loading machine learning, the model's weights into the GPU. And if we restart the application every time, like as we do like a development cycle, now that takes precious seconds here and there and that adds up. With Jupyter we can maintain the state of the program, the state of the algorithm and experiment on individual pieces without having to restart the whole thing, read the potentially gigabyte large weights file from the disk and loading those weights into the GPU memory.

[00:17:11] JM: Are there any frictions in the machine learning tooling stack that are frustrating you these days?

[00:17:19] DP: This question is a little harder for me to answer, because as an engineer, my mind always races to solutions. So that's probably I would make a great startup founder, because where I see the problem, I immediately see a solution using existing tools. So for me if there is friction, my default instinct is to see how I can overcome this friction. And barely oftentimes it doesn't register consciously for me.

I guess the whole transition from TensorFlow 1 to TensorFlow 2 was a little frustrating, because a lot of code bases do not transition smoothly. I know they have upgrade guides. And there is almost an automated way to upgrade the code from TensorFlow 1 to TensorFlow 2. But the internal workings of TensorFlow 2 is sufficiently different where code well optimized for TensorFlow 1 becomes very non-optimized for TensorFlow 2. But again, like that's not really a big issue for me personally or even for the team, because most models which we have been developing recently are in PyTorch. So that's not a big deal.

I guess the whole fact that machine learning development requires a GPU in many cases, that's a friction point. Again there is awesome tooling to facilitate with that and do cloud native development or launch a Jupyter Notebook in the cloud on the GPU machine, the development there. So we have solutions for this. One friction point is – And I'm sure like for everything that I'm talking about, there are probably companies out there that are working on solving this right now or even there are products on the market and I'm not aware. But just to iterate, let's say the weights for the models, these are huge files. And let's say if we want to version them, Git or GitHub is not great for that, because there are files which go into gigabytes. And yeah, we can use a large file storage in GitHub and manage those there, but it becomes very inefficient fast and breaks down easily, unless you're like super careful with Git LFS.

So what we end up doing is like very 90s solution, just a shared network file system in the cloud, which is expensive. So both AWS and GCP have similar offerings. In GCP, it's a file store. In AWS, it's Elastic file storage, I think that's how it's called, which offers a multi-client read and write functionality and we just store our model weights there. And in the context of Kubernetes cluster, we have this file system mounted as a persistent volume to every – The production pods.

I do intuitively feel that this is far from a perfect solution. And it feels brutal. It hasn't failed us so far. Specifically, it's not very important for us that the read and write IO speed is not great on these things, because we usually have to read the model weights once the end point starts up and loads the model weighs into the GPU. So that's not a big issue, but it's not an optimal solution.

Another problem that I feel have not been solved very well is – Which comes up when the training is data sets management. Again, at Rosebud AI we work with audio and video, and that means that our datasets are huge. Go into hundreds of gigabytes. And if you want to launch several experiments for training models, how to make sure that every experiment has fast access to the dataset. Because in machine learning, when you do an experiment, when you train a model, you have to go over your dataset multiple times. And that's how the model trains and optimizes. So how to ensure that it's cost efficient and how to ensure that a reading dataset multiple times of some storage is cost efficient and multiple parallel experiments using the same dataset do not compete on IO. And our solution so far has been based on putting dataset on the EBS drive or like similarly in GCP and just creating copies of that. Making a snapshot and creating copies of the cloud drive from the snapshot and each experiment having its own copy of dataset. It adds to the manual effort in launching experiments. But we have optimized that pretty well. But generally, I think it's not a very well solved problem, because you cannot use Elastic storage. Either Google Cloud Storage or S3 for large datasets, latency is high and throughput is low. And the GPUs will be idling most of the time if you put your video audio/data set there. So that's what comes to mind immediately about machine learning friction.

[00:22:36] JM: Do you think we could go deeper into some of the dynamics of the actual models? Like the generative adversarial networks and the mechanics behind training generative adversarial networks?

[00:22:48] DP: Sure. Let's do it.

[00:22:51] JM: Do GANs require a lot of data to train? What's the main challenge in working with GANs?

[00:22:56] DP: GANs do require a lot of data to train, much like any machine learning model. There is a recent research, and there are augmentation techniques that reduce the data load, but the whole space of machine learning is about converting data into algorithms pretty much. That's the whole promise of machine learning, right?

Generally, the models are not different in that regard. They need a lot of data. That is not the biggest difficulty I would say. So the reason why it's not the biggest difficulty is because generating models naturally are trained on unlabeled data. So the whole point of generating model is to find a way to represent a whole domain, a whole of space of things. So if you're talking about generating model that generates faces, the point of the model is to understand what the face is. And once the model understands that, you can generate realistically looking faces with sufficient variety and ways to control what's been generated.

But the generation process itself, because we want the generator model to understand what the domain is and how it looks, we don't need label data. So for generative purposes, getting datasets is in a sense much simpler than classical supervised machine learning where you train a classifier or even a reinforcement learning algorithm. Just get the data out there that you observe directly with zero labeling.

So let's say with faces, one of the most commonly used datasets for faces is CelebA, and I believe it's just human faces scraped from IMDB database, because IMDB website has a ton of headshots of actors. And that's a readily available dataset, or you can take shows from YouTube, slice them into frames and identify the frames which contain the face. And you have another dataset of faces. And if you want to, let's say, train the generator model to get a good representation of the face, like in 3D space, again, you take a YouTube show with a host and the host moves his or her head around, and that's how you get the frames of the same person, the same face in exactly the same setting from different angles.

So in that sense, obtaining datasets for generating models is easier. And that goes to the famous La Cake from Yann LeCun presentation where he's talking about how generative modeling is like the base of the cake. And the supervised machine learning is the frosting, and reinforced learning is the cherry on top. And he specifically refers to the availability of datasets. Like pretty much any data we have around can be used for generative modeling in the broad sense, including generative adversarial networks. And as a case in point, the GPT-3 is a generative model that is very good at generating fluent English, and not just English text, has been trained in unsupervised fashion on a dataset, which is huge. Has zero supervision and not very hard to build if you have resources.

So the difficulty in uh generative models training starts when you decide to use GANs. So GANs stand for generative adversarial networks. And the essence of GANs is that you have two neural networks. One tries to produce a fake another decides if the input is fake or real. And through the competition between the two, the generator learns how to generate realistic outputs and the discriminator essentially ends up assigning 50-50 probability. And that's the Holy Grail, the goal of training. Again, getting to the point where the discriminator cannot tell a fake from a real sample. And the generator can generate very realistic samples. And that's where the difficulty lies, because if you think about it, that's not stable equilibrium disturbance. But it is a very natural failure case in GANs where the generator learns to output just one very specific realistically looking sample.

So if you're talking about faces, one failure scenario would be the generator. Learning how to output every particular face, and discriminator learns that this particular phase, no matter how realistically looking it is, is always fake. And the training process breaks down from there. It's really hard to recover the training process once the generator stabilizes around one particular sample. And then there is a question of memorizing the dataset, where through discriminator generator can learn to memorize samples. Doesn't happen as often, but can also happen.

Generally with GANs, the training process is unstable. There are many hacks. There are whole guides and papers written about what a good hacks to make sure that GANs do not diverge. They're stable. But the hacks – There's no one silver bullet. I think if you're talking about GANs training, the most promising intuitive idea I've observed recently is originated in this community. There's a discord community. They are into generating fox faces for sonas. That's the word. And the hack they developed in this community is essentially reversing the loss. So at the point where either generator or discriminator becomes very good at what they're doing. They're kind of reversing the loss function and deliberately starting to make it less good, and that stabilizes the training quite a bit.

And another difficulty with GANS is even telling like how it's going, because unlike in supervised machine learning setting, there is no good metric to tell you if the training is proceeding well or not. The loss curves in GAN training misleading. So what was being used is different visual scores, which kind of objectively estimate the quality of the fakes. But that, again, just heuristics. They're not perfect scores. So when training GANs, what we end up doing is spending a lot of time just looking at the samples ourselves. Visualizing the samples, visualizing like different stages of the training process. Seeing how it progresses, and trying to infer from there if the training is converging, divergent, results getting better, worse. Are we stuck? Should we stop the experiment? Are be done training here? That becomes very subjective. There's no good hard rule for it.

[00:30:17] JM: Do you have to spend a lot of your time studying the cutting edge of research of artificial intelligence research? Or do you do you mostly spend your time on implementation?

[00:30:28] DP: I myself, I would say I spend most of my time on implementation. It better fits with my background. And like of course I probably read at least one white paper a week, maybe two or three. My goal is, again, like always practical. It's not about trying to see not just curiosity about what's out there, but my reading of white papers is always motivated by a specific product that we're working on or a specific challenge we have and trying to understand a collective experience out there. Because white paper, a good white paper summarizes a lot of experience that people – A team or research team has accumulated. And if it's written well, we can learn from it.

[00:31:20] JM: You use both AWS and GCP. Can you tell me more about the workloads and functions that you prefer for each of those cloud providers?

[00:31:29] DP: Sure. All our production workloads are in GCP. That was mostly historical random choice I would say. Part of the motivation was that for the business logic, for the applications that we built, we use Firebase, which we're big fans of, scales very well. And the tooling is great with Firebase. Helps us launch new applications and new features very quickly. And Firebase, being part of GCP, we can kind of get better network latency right out of the box. And in GCP we use Google Kubernetes engine, GKE. All our production machine learning endpoints run in a Kubernetes cluster in Google cloud. And for training, for uh training new models, that's very offline process in the sense but it doesn't have to be connected to the public in any way to the world. We use AWS.

And again, that's mostly historical choice. In AWS, we also use the Kubernetes cluster and launch our experiments in the Kubernetes cluster. So strictly speaking there would be much difference for us to move from one to another for either production purposes or experimentation purposes. We started doing that at some point and we continue that kind of separation. AWS for experiments and GCP for production.

As far as tooling in each cloud goes, I would say that at this point I have mild preference for GCP. I cannot put my finger on exactly what's better about the Google cloud. Maybe it is that the interface is slightly less confusing and the Google Cloud feels more like a single product. Unlike in AWS, you have a huge variety of how different screens and pages look depending on what service you're using. There is slightly less coherence in what AWS offers. The services don't – Well, they play nicely with each other mostly. GCP feels more of a single-minded product. But they're both honestly kind of a kitchen sink offerings, I believe, which kind of makes sense. They compete with each other and other cloud offerings. So we learned to use what we got.

[00:33:58] JM: Talking more about technologies head to head, how would you compare TensorFlow and PyTorch today?

[00:34:05] DP: It's like a VIM and EMACs debate of these days I feel like. Objectively we use PyTorch much more these days than TensorFlow. And many research teams still produce TensorFlow 1 models. So I would say we barely use TensorFlow 2, if at all. I cannot come up with a single TensorFlow 2 based model that we have in production or in a research phase. It's mostly PyTorch. My personal preference is it's really hard to tell. I kind of like the computational graph idea that TensorFlow represents more explicitly than PyTorch. But I think as a developer, as an engineer, I like PyTorch more. It just feels more natural, more imperative kind of development.

Yeah, as far as scaling, and like this more logistical aspects of productionize and machine learning research, I would say there isn't much difference between the two. They're both fairly straightforward to productize. I think in our case, and I'm sure for many other teams, it boils down to existing open source research that we want to understand and potentially reuse parts of it what it was written in, if it was in TensorFlow or PyTorch. And most of the research these days comes out in PyTorch. So that's part of the reason why we are using PyTorch more.

[00:35:45] JM: Image generation is still a developing field, so is video generation. There're a lot of new architectures. There're new techniques being developed in the research community, but what do you see in the near future?

[00:35:58] DP: What we are seeing like objectively, and I touched on this point briefly, controlling existing models. So much like with GPT-3, what I expect to see in the future more is big budget teams training large models, which will exceed the performance of existing models and like blow our minds. And we will take these models and we will learn to control them much more. We will understand what's going on under the hood. And without training completely new ways of representing audio or video or text, we will learn how to control it more. And even with the GPT-3, we're seeing like the whole teams focusing on how a prompt can be formulated to achieve a certain level of certain kind of generation and making GPT-3 solve a particular kind of problem, which it was not designed to do. So that's what I see in the space of other generator models as well.

So if you're talking about images, I expect to see probably an evolution of BigGAN, because BigGAN is unique compared to, say, StyleGAN. StyleGAN is awesome at generating realistically looking objects from a particular domain, say, faces, landscapes, animals, cars. But then StyleGAN is single domain architecture, while BigGAN was successful in generating multi-domain images. So the same model essentially learned how to represent a wider world. So what I expect to see is more models like BigGAN, which can represent a wider variety of visuals at high fidelity. Probably some kind of tradeoffs between the fidelity of generation and the speed of generation, and hopefully we and the rest of the community will be able to have access to those models and learn how to control the outputs that these models produce. Learn to control the models. And through that, build products in which we can leverage this model in generating persuasive visuals.

[00:38:14] JM: Can you tell me more about products that you anticipate building in Rosebud?

[00:38:20] DP: Yeah. I can probably start with a vision for the company, which is enabling everyone in many different situations to express their creativity and create art essentially. So our future products likely in the nearest future will all be focused on video generation. We have the talking heads product and we are actively working on adding features to it and extending what it can do. Again, going back to the question of controllability, making sure that we can control what's being generated and surfacing those controls to the user.

So from the perspective of what the end user would see around along that dimension, in the nearest future I expect Rosebud AI to launch more products in the video generation space. As far as technology problems and challenges that we are going to solve and solving right now, they are all centered about how to control powerful generated models and how to surface those controls to the user. And that includes not just technical challenges, not just fundamental research challenges, but also the UX challenges. Because if you have 500 levers, you cannot expose all of them to the end user, that will not make sense. So there is a big question of good design of the applications that we build.

I guess, overall, moving in the direction of creating art. So one helpful way I think about it is you have a spectrum. On one end of the spectrum, you have, say, photorealistic virtual reality experience. So you put a person into a completely realistic environment and there is like zero art in it. Let's say we can capture with highest possible fidelity, cathedral somewhere, and I put the VR headset on and I can see this cathedral and every little detail of it and there is zero art in that because we're just replicating the reality to the highest degree possible. And then if you're thinking about a picture, you take, say, with your phone, you are still working within the base of the real world, what the real world has to offer as far as visuals go. But you have control over frame and lighting and which moment exactly you capture. So there is a way for you to put in some artistic intention into the photo.

And then let's say you have a collage and you can combine completely different unrelated visuals into one art piece and something that would not even coexist in the real world, and through that you can tell a story or can be an emotion. And like on the complete end of the spectrum, you have like the paintings where everything is fictional, every single piece of it. So the products that we will be launching at Rosebud AI are towards this end of the spectrum where anyone can start with some realistically looking and maybe pre-existing visuals about the real world, but essentially be able to tell a story or convey an emotion while not being an artist themselves potentially. So we want to facilitate that process, where if I – Let's say like as an example, if I have a story in my head and I want to tell you this story in a visual way, how do I do it? These days it's really hard for me to do, because I'm not a painter myself. I don't have the budget to hire actors and the director and the writer to get a script and shoot a movie. I just want to tell you a story. So what is the way for me to do that? There is no easy way to do it. What we build at Rosebud AI tools of expression that would allow me to tell the story exactly as I see it in my head, especially transfer's what in my head with a minimal effort on my side and communicate this to the world. That's very ambitious, and I do expect that a large part of that is not just machine learning specifically or generating models and the difficulty of controlling them and training them, but also in the user experience, because if I'm not a technical user, what kind of levers should I have for a powerful tool like that to tell my story?

[00:42:50] JM: Well, that seems like a good place to close off. Dzmitry, thanks for coming on the show.

[00:42:53] DP: Thank you. It was great.

[END]