(no title)

77 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=26FTD08ZPOU
https://www.youtube.com/watch?v=26FtD08ZpOU

SUMMARY

The lecture provides an overview of multimodal models, emphasizing their significance in integrating various data types such as text, images, audio, and video. The discussion highlights the evolution of models like CLIP and its successors, which leverage transformers for encoding and decoding different modalities, ultimately aiming for an "omnimodel" that can seamlessly process and generate outputs across all modalities.

- *Multimodal models integrate various data types (text, images, audio, video) for comprehensive understanding and generation.*
- *The goal is to create an "omnimodel" capable of handling any combination of modalities.*
- *Transformers are the backbone of these models, originally designed for text but adapted for other data types through tokenization.*
- *CLIP (Contrastive Language-Image Pre-training) serves as a foundational model, aligning images and text through contrastive learning.*
- *Subsequent models like Cichlip and Lava enhance efficiency and capabilities, allowing for better handling of multimodal data.*
- *The Quen series of models further improves on previous architectures, focusing on dynamic resolution and long context understanding.*
- *Challenges include managing the differences in information density across modalities and ensuring stable training when integrating diverse data types.*
- *Recent approaches like Chameleon explore mapping all modalities into discrete tokens, although this can lead to information loss and training instability.*

So um originally the plan was to talk more about reinforcement learning but I thought since I only had one lecture to give I think this class would be a little bit incomplete without saying something about multimodality which is so pervasive if you look at all the major models. Um so this will be a overview of um multimodal models. Um, of course, this could be a whole class in itself. Um, and I'll try to do what I can in one lecture. Okay, so let's dive into multimodal models. So, so far in this class, we've have exclusively focused on language models. And language models are already pretty general. They give you the ability to go from any piece of text to any text. So this could be any language uh code um you know poetry any anything you like even you know let's say uh DNA for example um but the world is multimodal um you have text you have images audio video and if you think about a sort of a north star of where we want to be is what people call an omnimodel which is ability to take any combination of any of these modalities ities and output any combination of any of these modalities.

So for example, I can give you an image and a video and I can ask a question about you know which uh about the image and the video. I can generate images. I can convert the audio into an image. You can imagine all sorts of different applications um by having a omni model. So, you know, in this class, I'm not going to build up to a

whole omni model, but I'll give you um you know, some of the pieces that are um at least allow you to ingest a little bit of um of images um which we'll mostly focus on today.

So, so how do you build such a thing? Well, um this is sort of a a weird way to motivate this, but um the reality is that transformers work really well and despite the best efforts of people to try other things um those are in across all these modalities still at scale the the best thing we have. So we have to figure out how to use them. And now transformers were designed for text. So they have this property that they speak you know tokens. So they take input as input a bunch of tokens and then output some tokens. And here I'm um sort of expanding the notion of token to be not just a discrete token that we've seen in text but also uh potentially continuous tokens. You can think about these as embedding of uh tokens.

And the key thing is that a token should represent some sort of semantic unit of information. Um for example in natural language uh tokens are subwords and these are you know somewhat meaningful whereas for example a pixel is is certainly not uh meaningful by itself. Um so somehow we must convert everything including you know audio and um and images into either discrete or continuous uh tokens.

Now notice that we've already had to do this even with text right in the first lecture we talked about tokenization and there it was um you know not too bad I mean we had a BP tokenizer you guys implemented it um it was you know it's fine um one could wish for um you know a better tokenizer or no token uh something that doesn't have tokenizers at all but um it's not the worst thing in the world. Um now for non-ext modalities um we have to scratch our heads a lot more and figure out what is the equivalent of the BP tokenizer that will take an image and produce things that a transformer can digest.

Okay. So you have to answer two questions here. One is for non-ext data for example images or videos or audio how do you input these into the transformer and then how do you generate them out and this lecture will mostly focus on uh question number one. Okay. So so let's start um back in rewind the clock back to uh 20 21. Um so the clip model which some of you might be familiar with is a lot of the foundation of uh modern you know uh VLMs and so we can go through this a bit uh more in more detail. So, clip stands for contrastive language image pre-training and a bit of historical context is around that time um you know com you know GPD3 had already happened GP2 all the language models had already um gone into this kind of foundation model era um you know vision was still um you know there were some efforts but you know traditionally had been based on large annotated um data sets such as imageet and training models um such as the ResNet on these data sets and doing a bunch of other things like data augmentation and getting really good performance.

And so the researchers at OpenAI were wondering is it possible to leverage the large amount of image and textual captions um out there? So inspired by this idea in language where you could just scrape the internet you have all this text it's very noisy but somehow if you train a large enough model you can make sense out of this and generate something useful. Um so what was the equivalent of that for for images? So as a result clip was born and the idea of clip is uh fairly you know simple and straightforward. So the idea here is that suppose I give you a bunch of image text pairs. So let's say I give you 32,000 of these. Okay. So what I'm going to do is for each image I'm going to encode it using image encoder which I'll explain later into a vector. Um so I'm going to have

third uh 32,000 of these um image encodings.

I'm going to also do the same thing with the corresponding uh text. So that'll give me uh t1 through tn. Right? So these are the embeddings of the text. These are embedding of the images. Now the objective I want to set up is that um I want the you know image let's take I1. So I1 is associated with aligned with t1 right. So I want this um alignment um the dotproduct between these two embeddings to be much larger than the dotproduct between I1 and all the other texts. Okay. And conversely I also want um for this text to say that this dot productduct is larger than the dotproduct between T1 and all the other images.

So that in a nutshell is the clip objective and you can imagine doing there's um basically two to two * n um different um kind of softmax classification um you know problems. So here's um here's the here's the code. So you get a m byd matrix for the image encodings and m byd times for the textual embeddings. You normalize um and you take the stop product um with some with some temperature and then you compute the cross um entropy.

Okay. So it's basically like a multiclass classification problem where the examples are sort of structured in this kind of um you know n byn matrix form. Okay. Any questions about that? All right. So so this um so some more details about how clip was built. Um so where does the data come from? Um there are not you know too many details in the in the paper about uh the the data. Roughly it's um you take a bunch of queries um uh from somewhere and then you go search online for these and you mine a bunch of image and text uh pairs and um this resulted in 400 million text image text pairs. um the data set wasn't released but there was an effort um by this paper called you know open clip which uh replicated and and further kind of extended the clip ideas um um so there was this data set called lion 5B which was released five billion you know images with textual descriptions and there was a bunch of processing that happened um interestingly they actually used clip for the data filtering and then trained open clip. Um so you know there's some bootstrapping happening but at least this is a model where you can point to the data set that it was trained on and um the code as well.

Okay. So, a few other you know details. Um, so images come in all sorts of resolutions, right? If you find an internet image, it might be uh long and skinny or tall um sorry, long and skinny or the other way. So, arbitrary dimensions. And um one thing that you learn about neuronets is that they don't like things to be dynamic. they want things to be you know fixed size. So there is a sort of somewhat um you know heristic uh processing that happens is they resize the image so that the shorter side is 336 or whatever target size could be 224 as well. Um and then they center crop um to get a square.

Okay, this is obviously for convenience. Later we'll see that you can do much better than this but this is just for expediency and also at this time people were the clip authors were thinking about image net and classification. So um usually the object is in the middle and you're just trimming off some background so it doesn't matter too much. Okay. So um so what is the vision encoder? So this is the general algorithm but I've uh haven't specified what the image and text encoders are.

>> Yeah and text when they input right. So they provide text every type of email generation that we follow will

have some sort of it. That's why they train on both text and image rather than just images. So the question I think is why they train out image text pairs as opposed to images. >> Right? So there is a line of work that tries to encode images um into representations um based on data augmentation. So for example, if you take an image and I think this idea called sim clear, you augment it, um you crop it, you randomly perturb it, um rotate it and you want to say that that is the same image like they should have similar embeddings. Um now the and that's that's useful for kind of low-level details but um the problem is that um you won't data augment your way from like one type of dog to another dog. So by using that text it gives you higher level semantic um um representations of the images and yeah that's a good question and later we'll see how these semantic representations are useful if you look at the types of tasks that people are trying to uh solve.

Okay so what is the vision encoder that clip uses? Um they actually experimented with a bunch of different ones um resnats and vision transformers uh which had just come out and they found out the vision transformers perform the best. So when people say clip they usually mean the vit uh version and so you know a vision transformer does the following. Um so you take an image and um you you break it up into these patches. Um I think the original paper was like 16 by 16. Um and uh the clip used 14 by 14. Um but anyway you have these little you know patches which are basically you know vectors. So each patch is in some sense a a well a token for this you know vision transformer.

um you add the positional embeddings just like you would if you're training a a language model and then you just put it through a standard transformer. Um and if you want um to you know classify then you you would um well you would usually want to um okay so the trans sorry the transformer encoder produces a a a sequence of vectors right and um the clip paper does something a little bit different you can average all these vectors that gives you a single vector u but they saw it was better to do a attention pooling which means that you take the global average of all the activations but then you also do another round of uh um you know attention with that query against the key and values of each position and that gives you another vector which is maybe a little bit more um informed rather than just like a a straightup average.

Okay, so that's uh the vit and um the clip paper found that the best model is what is called vit l14. Um so this to decipher this this is a large vit um I think it has like 24 layers or so um not quite sure about that. um and 14x4 patches. Um each patch is RGB, so it has three channels and um they trained on 336 by 336 resolution images. Actually, they trained on lower res and then they um up you know uh and then I guess for speed because high resolution images takes uh longer and then for the latter part of training they train on higher res. Yeah.

>> Yeah. for the position embedding would it be more complicated than text for example X only has like one dimension but even has two output down >> yeah so a question is um can you do be smarter about the position embeddings because this is just you know linear zero through nine um I think in this paper they actually in the clip paper I believe they tried some 2D version of this and the 1DNA they found that it doesn't really that matter that much.

Um, I think it's also, you know, I always have to take these results with a with the idea that they're interesting classification in mind. So, for classification, maybe it doesn't matter. Later, we'll see fancier positional

embeddings that do take into account the um the spatial structure. Okay. So the text encoder um is a standard transformer. It's a GPT2 style transformer since um that was from the same group that developed GPT2.

And um in order to get a single vector out of a sequence um they prepended BOS and um appended EOS and then they took EOS um the activation at the highest layer as the um the representation of that whole sequence. Okay. Okay, so you have the vision encoder, you have the text encoder, and then you can do this um process to to train. Uh you pick a batch, you encode all the text, you code all the images, and then you form these um two end cross entropy losses and then you go from there.

Okay, so the headline result and which got a lot of people excited about clip back in 2021 is that on the imageet benchmark, zero shot clip outperformed a resnet that was trained on 1.2 million imageet images. So if you think about this 1.2 to image net images. This was many many hours of Amazon Mechanical Turk worker time to do this annotation. Um, and now you have clip that was trained on more organic you know web data. Um, of course this was the labor of a bunch of people on the web but you know if you're able to leverage that or existing data already then you can just zero it. And the zeroot technique is basically um to to explain I don't have slide support here but you take an image and then you have a bunch of text and then you basically take the dot product with the various labels and see which one's the highest.

Oh yeah it is. Sorry it is on this part of the site. Okay good point. So this is a zero shop uh prediction um where you take an image and then you Yeah, what I said. >> Cool. Any questions? >> Yeah. this type of sound supervised version is actually an issue if uh for example there's a dog image like for all the caption but maybe not just this simple this dog maybe other caption also would that confuse the model >> yeah so the question is if you have a dog and there's maybe other captions that also have dogs would that confuse the model um in general this process is going to be noisy um I it's okay if there's another dog because on average um it's unlikely that there's always going to be a dog. Sometimes it's going to be apple or a cat or something.

Um but but also another point is that these images are um basically scraped from images on the web with textual text next to them or in the alt um kind of a image. And so they're extremely noisy. There's been, you know, studies that look at when you have a caption of a of a image, it doesn't necessarily just like verbatim say what's in the image, right? Because if you have an image and it's a dog, you don't need to say a dog.

It's um so this is process is very noisy. So it's um maybe uh kind of somewhat surprising but also interesting that it worked. I mean there was a lot of data filtering that is needed. So if you take arbitrary images on text on that web, it's probably going to be too way too noisy. One other thing to note which um relates to a point I'll come back to at the very end is they also try this um alternative where instead of doing this ranking, they can predict text from the images.

So they can set an objective where they try to take the images and predict the text um either as a bag of words or um as a language model and they show that you know maybe I don't know somewhat surprisingly that if you use a stronger model it actually does does worse or at least is less efficient um compared to using bag of words um you know clip. So this is saying that this is you know I I think saying that for the actually the representations

that uh you're trying to learn or again this is image net accuracy so it's classification um actually modeling the exact token sequences of of the caption isn't um so important for getting the rough um representation of the image.

Okay, we'll come back to this uh point at the at the end. Um, so so what do we learned so far? Um, so we've encoded images using clip and it captures the semantics of the image because it's paired with text and generally text talks about the the semantics. Um, it is important to note that the design decisions here are based on image classification. So it's not very, you know, fine grained. Um, but yet it still, as we'll say, serves a as a kind of robust starting point for everything we're going to do later.

One technical kind of downside of clip is that it does require large uh, you know, batch sizes like 30,000. Um, if if you have a batch size of one, clearly it doesn't work or even 10, it doesn't work. And furthermore, the softmax operation operates over the full batch. So it's not really very decomposible. Whereas if you think about normal language model training um a batch of examples all the sequences kind of paralyze and you just like do a um a aggregation at the at the end.

So this point will be addressed by another uh paper um from Google that uh came out called cichlip. And so think about this as basically an improved version of clip. So it stands for sigmoid loss for language image pre-training. Um and here the the main difference is that clip does multiclass classification. It says I have this aligned text and image and I'm going to say this is positive and against all the other alternative images or against all the alternative texts. cichlip in some sense is a lot simpler. It basically says for any given image text pair are they aligned or not? Right? So if I reference this diagram up here, it's basically binary classification where the diagonals are positive examples and the off diagonal are negative examples.

That's it. So so here's the algorithm. um you take the embeddings, you normalize, you take the dotproduct as usual uh but now you're uh doing um you're forming the labels which are minus one on off diagonal and diagonals you have one um and then you do a um you know log sigmoid. So the objective is very simple. Yeah. little first as the sampling structure because I think most of the samples are not quite so random cycling the laws I think will >> yeah so the question is does this require any sophisticated sampling uh strategies um at least in the initial paper they were just operating on literally the same type of matrix um you could imagine I You're probably thinking in general for kind of these contrastive methods like the negative examples need to be sometimes you need to balance the positives and the negatives or the negatives need to be like tight negatives so that you're not biased but um at least in the initial version this was you know fairly just straightforward.

So um to speak about the data um you know uh this was done at Google this was a data set they used another paper which was another image language image model around 2022 which got a web li data set with you know order billion image text pairs um they also do did some extra work for example image that had text in them they did OCR and that's another way to form text image pairs and did some filtering ing this was multilingual. Um the main um po you know uh point of this this paper was that it was much more efficient to train than than clip. Um so clip was trained uh 10 days on 256 uh TPU v3s and cichlip was 5 days on 32 v4s and you might think oh v4s are actually should be faster than v3s in terms of flops per second they're actually not.

um they're better because you can put more of them in a pod and the interconnect is better. But um at this scale it's really um actually you know not um not faster actually you know uh like 60% slower or something. Um so the the the way that this is uh you know faster and of course I think clip probably did not try to optimize the the code for the maximum you know throughput. it was just kind of let's get this thing to work. Um but the cig paper showed that you can um you know parallelize this as follows. So you have basically and if you recall your kind of systems um lecture. So this is can think about it as you know DDP if if you if you like where each device is um uh stores a subset of the the you know image text pairs. Um however there's interactions between the examples unlike in language model training where everything just factors. So in the first uh pass um each device just computes all the losses on um on the its whatever image uh text pairs it has locally um and and then and then you you send um the text. So basically device one gets uh T5 through T8 and now it's able to compute the negatives here and then it gets uh T9 through T12. So you kind of rotate around until you cover all the the off diagonal uh block entries.

Um another thing that uh they was nice about cichlip is that you effectively decoupling batch size from the loss. Um whereas in clip the loss is tied to the batch size. If you change the batch size it's a different loss function. Um so they were able to experiment with much smaller batch sizes um less than you know 16k and in that regime um it's much better than clip because the loss function if you have too many too uh small of a batch size is just degrades whereas um for smaller batch size for cichlip um you have more variance but um but it's the same in expectation is the same loss. Um they also say wow you can go up to large much larger batch sizes but that doesn't really help. Um and we saw that critical batch sizes uh at some limit and 32k was their essentially their critical batch size.

Okay. So now we have clip and cig. Um these are image encoders that take an image um in this version just a fixed size image 336 x 336 let's say and maps it into a vector okay which uh contains um some you know semantics. So now let's start building um um VLMs, vision language models. And I'm going to talk about two families of models um Lava and Quen. Um the two are very similar in the sort of broad template.

There are some details that are different which we'll we'll go through. And the basic idea is that we're going to take these embeddings and then just inject them into a language model. So this is going to be more of a flavor of sort of mid-training or post-training where uh we take an existing image encoder, we take an existing LLM and then we're going to kind of stitch it together rather than training something from scratch. Okay. So the LA lava paper came out in 2023.

Um and this got people excited because um at around this time all the you know the closed models um like GPD I think it was GPD4 was able to do kind of visual reasoning and um they were able to show that they could do some visual reasoning as well. It wasn't as good as GPD4 of course uh but um but you know I think it was an open model and people got to see what went under the hood.

So um for VLMs there's essentially uh a few pieces here. Um there's a vision encoder. They use clip here. For the text decoder they use a language model called Vunia which is the the first llama model that was fine-tuned on some shared GPT conversations. So these are uh conversations that people had with chat GPT and shared them

on this website. I don't think this is up anymore. Um so the data that they used was um based on um MS uh Coco. So this was an annotated data set where they had mechanical Turk workers go and annotate a bunch of images with bounding boxes and captions describing what was in the image. Um so they basically synthesized a a data set for their model. So they prompted GPT4 with the captions or the detected objects and asked GPT4 to generate questions or conversations. Um for example um this is an image in the MSCO data set. This was an annotated um caption done by a human and also uh here is the you know also annotated by a human. Um and then you ask GPD4 generate uh me a conversation and you might get a a question and an answer. Uh you might ask GPD4 generate me a detailed description. It generates something I guess similar to the caption and then you can ask GP4 to generate complex reasoning and it does um you know something like that.

Okay. So this is syn syn synthesized uh data 158,000 examples um and and then they you know trained a model on this. So the model um so they use they use clip um in particular the the highest performing clip uh version VITL14 um so 14x4 patches um and then they let's just look at this image um so they take images they send it through this vision encoder clip that gives you a vector but this vector isn't really in the same space so to speak as the text. So then they um multiply by matrix W to get another vector which is in the same space as the texture embeddings. And for text you basically just use your standard embeddings to get vectors.

So what's basically happening is that the text gets encoded into these vectors and the image also gets encoded into these these vectors and then these vectors this whole sequence of vectors just goes through a standard transformer and produces output. Okay, so we're in some sense converting these images into textual tokens um so I can leverage the pre-trained uh language model. Um so to train this model um they do uh it two in two stages. The first stage is that they they freeze the vision encoder and the language model and only train this matrix W. And this is what they call the alignment phase which is essentially making sure that the the image um gets mapped into kind of the same you know space because if you just use a random W um these these images are not going to be um I mean certainly they're not embeddings representing any sort of like natural language token. So the goal of training W is that they sort of look like natural language token embeddings.

And the second stage is that they now still freeze the vision encoder and then train W and the language model. So they're fine-tuning the language model on on um their their data. So remember their data is basically here's an image and here is a conversation or description or complex uh you know reasoning example. So this is all image plus text to text. Okay. So um this is one example from their paper. Uh it's kind of interesting. So you have an input image and um a conversation. The user says what's unusual about this image? And their model is able to tell you that uh you don't usually iron on the back of a minivan. Um, and they also make a point that their model is is pretty good because even if your your user um uh prompt is not really prompting like the unusualness, it still talks about it whereas um you know GP4 obviously is able to do this but uh other models at the time uh were not um able to.

Okay, any questions about lava? Okay, so that was 2023. Um, and I'm going to sort of skip forward to uh Lava 1 vision, which is 2024. There was a sequence of papers after the Lava paper came out of Lava 1.5, Lava Next. Um, and the main but but I'll capture all of these innovations I think in what I'll describe next. Um the main thing that they did is it's essentially the same recipe but they were trying to be more ambitious in terms of the

types of multimodal applications they could do.

So they um were able to handle multiple images and videos. So here's the same uh sort of diagram. Now they can take an image, a single image or multiple images or a video which is essentially a sequence of images um corresponding to a sampling of the frames. Um and here the vision encoder they upgraded to cichlip. Um the text decoder they use Quen 2 now which was um probably the best uh language model um that was out at that time. And then for the uh projector or the adapter which is the thing that takes the output of the vision encoder and turns it into the input of the the language model is a two-layer MLP instead of a linear projection.

Okay. So relatively you know it's like you have a system and you're just like upgrading the parts but it's the same kind of rough system. Um so the let's talk a little bit about data processing. Um so one thing that they wanted to do is to do OCR and the thing with OCR is that you need to preserve very fine grain information. Um otherwise a J looks like a I and that's that's not good. Um so remember that clip resizes and crops to 336 by 336. And clearly if you have a document and you crop to 336 x36 you can't read it. So uh their solution is this idea called any res uh which was actually introduced in the lava 1.5 paper and the idea is basically um I mean it's fairly straightforward. you break up an image into, you know, multiple, you know, pieces. Um, and the idea is that each piece is the resolution of whatever the vision encoder is going to take, 336 x 336, for example. So, you're going to code all the pieces and then you can into a bunch of vectors and then you concatenate the vectors. Okay? So, you're you're basically noticing that your vision encoder can't handle high resolution. So rather than downsampling, you're just going to uh crop and look at different parts of the image.

And now if you have really super high resolution images or videos, then you get too many tokens. So then they have to then they sort of down uh sample. But um the idea behind any res is that it's sort of adaptive. And this is sort of nice because the transformer is already adaptive. Language can be your sentences can be any length. And the transformer already handles that pretty well. So it turns out images can be any resolution.

So that's kind of picking back on the same um kind of dynamic uh ability. Um so here is uh what they actually do for for images here. So let's say you have this image of a of a paper. Um they do down sample and have one path which is basically a down sampled encoding of the entire paper. But then they break it up into these chunks where each chunk is the resolution that the vision encoder expects and then encodes each of these separately. And then you concatenate all these patches and if you have too many then you sort of you know reduce that the number of patches and by um interpolation.

So okay so then they're able to handle images multiple images and video. So you think about well you know all of these are technically all reducible to images but they put their thumb on a scale a little bit because um they want to make sure all the modalities are kind of roughly uh you know comparable um because videos can be very long. They don't want their data set to be dominated by a bunch of like repetitive you know frames. So here's what they do here. So for a single image they basically have the full image downsampled and then a bunch of crops um up to n crops there up to nine. And if you are presented with multiple images then they just uh use the sort of the you know the base base resolution. So if you have a single image I get to look at it more carefully. If I have multiple images I'm just going to look at it kind of from um you know afar. And for video um we're

going to use um even kind of lower resolution or fewer tokens to represent each frame because the idea is that well videos can be um you know they only use up to 32 frames. Um uh but you know the the you start running into kind of context length uh problems uh for video and later we'll see how you know a big part of being able to handle multimodal is is to deal with long context.

Okay. So um the data here that they use is um is I would say that they state the philosophy is like they want to curate high quality data and quantity. Another way to interpret this is that it's very targeted uh data. If you look at many of these it's like very taskbased whether you're doing visual question answering or um you know answering questions about tables. So it this is definitely kind of post-training you know territory where you're you're trying to get your model to do these tasks and therefore you create a bunch of these tasks and this this work is also um unabashedly basically you know distilling GPT4 um you know models so that they can get the best you know performance which is um I guess not ideal but um this is what uh you do if you don't have a annotation budget. Um and uh so they have single images, multiple images and videos. So there's a lot of different um you know uh you know pieces. Some of these are very specific like you give two images and you're trying to spot the difference. Um but one thing that uh okay I'll talk about I'll come back to this at this point later. So for training it's um roughly the same idea as before where the first stage is that you're focused on uh alignment.

So um you know in particular you only train you know the projector which is the adapter. Um and then in the second stage you train and the third stage is you train the full model. And before we had two stages, now we have three stages. Um I think you know I'm not sure there's any particular principle reason for this except for the second stage is trying to put high quality data but more focusing on knowledge and um the final stage is uh focusing on examples that look like your downstream tasks.

Okay. So one thing that they um found in this paper which was kind of interesting is um that you get transfer between modalities. So even though they have only single image data for diagrams and charts but this actually generalizes to the multiple images. So at test time it's ne at training time it never saw um example where you have a table and a chart and you're asking questions about it. But at test time, you have a t um you have two images, one for uh the the table and one for the chart, and you're able to have some sort of conversation about it. um at training time you only have OCR data on single image data and you only have relational reasoning for multiple images but you can generalize to these uh cases where um which are useful for powering guey agents where you show it a bunch of you know screenshots this is multiple images and it can generate um a description of the the the screenshots.

Um another example is that um they have this idea of visual prompting u where uh you have this circle which is a part of the image but that's meant to highlight the part of the image that you're supposed to you know do something about. Um and this type of data only exists for single images and now it can generalize to um you know videos where the user says describe the player highlighted in the video and this player is across multiple frames.

So this is you know when I first look at this um and I said oh boy you know you're basically targeting each of

these tasks. is kind of like supervised learning. But um but if you have enough tasks, these models seem to do do some transfer um which is which is uh I guess reassuring. Okay, so again this is a fairly standard uh VLM template vision encoder plus the projector um into the language model. Um there's a lot of work that goes into the the data curation. and they lean pretty heavily into synthesize task specific data. Um the nice thing about the Lava series is that it it's one of the few works that open sources. They're not just the model weights but also the data. So you can really replicate and study this stuff.

Okay. Now let's talk about uh Quen models. Um so Quend started also training in 2023 multimodal models. So the first version was QuenVL and uh hopefully I'll go through this quickly because you hopefully see the pattern. The vision encoder they used open clip um remember open clip was a open reproductions of of a clip model. Um so it's basically a clip um encoder. Um and they for the adapter um they did one layer of cross attention incorporate 2D positional embeddings and maps to a fixed size of you know uh 256 which is a little bit you know I I guess uh I guess this will be changed later um because this is definitely not very dynamic but neither is the vision encoder at this point. Um I guess this got caught off but uh um but uh the special tokens are like they have an image tag and a box tag for bounding box and a ref tag for um description.

This is what happens when your slides are in HTML. Um so for training they have three stages just like with the lava models. In the first stage, um the idea is to I mean they call it pre-training but it's really you know you're not pre-training from scratch. Um they uh this is generally large scale lowquality data. They freeze the um the language model and train the vision encoder and adapter. Um so this is a bit different in that from um the lava models in that they train the vision encoder um and at this point um and you can see some of the data sets that they um are are training on 1.4 for you know billion examples. Stage two they move to higher quality task specific data and you can see some of the kind of the usual suspects of like VQA data sets um you know chart question answering and so on. Um and then they train all the parameters here. And finally stage three is um you uh instruction tuning data. Um you freeze the visual encoder and you train um the the adapter in the language model. Um here are some examples of the type of things that they're able to do.

So um of course this is Quen. So may some of these examples are in in in Chinese. um their interest is also making a model that is supposed to be sometimes subsuming a language model. So a language model can do code. So we want the vision model to be good at code as well. Um and the you can have this example if you can see it is an image and can fight Spider-Man the Hulk and it can output bounding boxes um you know as well. It doesn't actually output an image. It just outputs um uh you know a bounding box.

Um and then you can do OCR. You can compare um you know. So use your stuff. Um okay. So then Quinn 2 um was um again a kind of upgrade. There's some new ideas here which I'll go through. Um so they use a larger vision encoder. Um they also started doing uh dynamic resolution. I think this was the main thing we remember when we went from lava to lava one vision they realized oh we need to handle images and of different sizes and if you try to do video you it's clear that you need some sort of dynamic resolution. So um the model is is basically so let's say you have this this uh picture here that uh might be mapped to um 11,000 tokens. This tiny picture of a equation might be only mapped to eight tokens um and so on and so forth. Um so how do they um you know do this? Well, basically each um it's a kind of the same idea of the NRA idea from lava where each uh

224x 224 patch is encoded with a vit um and remember the vit that they started with is the open clip vit but this gets um you know uh fine-tuned um and then they try trying to reduce the context size they uh compress every 2x two into one. So that they ultimately every patch uh gets um generates 66 tokens for videos. They sample two frames a second but you know max out at 16,000 you know tokens. Um um okay so one thing coming back to this question about positional embeddings um so they did do something different uh here with Quen 2. Um so they use this idea called multimodal rotary uh multimodal rope. So remember we saw rope before and the idea behind rope is that you have a um you know embedding such that the um you know the inner product between the vectors kind of depends only on the kind of distance. So the distance is defined in 1D by just the number of tokens you're away from it. So now uh the multi-dimensional version is essentially the same thing except for now you have um 3D. So you have height, you have width and you have time. So each position, each patch now is a a triple uh defined by the the coordinates. And then to compute the M rope, you basically for every dimension you compute the rope and then you concatenate. Okay, so it's a fairly, you know, straightforward um idea. Although later with Quen 3, we'll see how this is actually um a bit sub-optimal.

Um okay, so they initialize the language model with, you know, this is the Quenv model. So of course they're initializing with Quen. Um and then they have three stages of training um which is very similar to um what bef we did before. And then they were able to show a bunch of you know different capabilities um you know some video understanding um you know it can do math and code function calling um and and so on so forth.

Okay. So quickly I'll talk about Quen 3. Um so again there's a few kind of changes here. Um I wouldn't say these are kind of structural big changes but they are probably changes that do impact the the quality of the model. So this is from last year Quen 3VL report. Um and so it's a roughly the same uh diagram here. I'll point out a you know maybe a I'll come back to this diagram. I'll um go through some of the changes. So one is that now they're using quen 3 models.

They have a a series of dense ande models and the quen 3 models are you know really really good and it really helps the quality of the final model. Um one thing that they are invested in is long context understanding. So their context length can go up to 256K which is uh really important if you're trying to do long video. Um let's talk about the vision encoder. So they use cichlip 2 which is basically improved version of cichlip. It's actually an identical architecture is designed to be backward compatible. Um and then um they improved their rope and um the way to explain this is before you had the three dimensions time, width and height and uh basically uh going to ba um basically allocate the first block of uh dimensions to the time then the next block to width and the next to height. And the problem if you remember um rope is that you know each uh component refers represents a different you know frequency. So this would mean that maybe all the temporal dimensions are low frequency and all the height dimensions are high frequency. So they basically interle them um when they realize that and they now um all the the axes are exposed to both low and high frequency.

Um another thing they did was explicit video uh timestamps. Um so if you look at closely at this uh sorry this was wait okay this is quant three. Oops. Um so if you look uh this is probably a little bit too small but um before the time stamp was kind of implicit in the the positional encodings right each frame in a video intrinsically gets a different you know uh there's a notion of time just because it's in a different position. Um but here they made

the time explicit. So these are actually token 0 seconds is a token that is is now you have a token that represents the time.

Um and they found this uh you know to be helpful presumably because it's something that you can directly refer to like what happened at you know after two seconds. Um so that is the main uh one of the main changes. um they also did this square root normalized per token loss. So the idea is that you know some examples that call video are very long and some examples that involve a single image are very short and the normal thing is that every token is treated the same and this would mean that the video examples are going to you know you know dominate. Uh so they basically um the details aren't too clear but I I believe they um normalized each example by the uh square root of the um the length uh to basically downweight the impact of really long examples.

Um the the final thing that is uh worth noting which is interesting is that they did something interesting with adapter. So remember we start the adapter is a thing that connects the vision encoder to the language model and lava was the simplest. It was just a linear projection and then we got an MLP and we got some cross attention and uh deep stack which is actually a paper from the deepseek uh team u but when you know used here is uh is more sophisticated.

um they noticed that well the vision coder already computes a stack of um you know vision uh you know embeddings and we're basically going to basically add these directly into the residual stream of of the um the language model. Okay. So this is sort of a bit more of a kind of a deep fusion of the the vision um encoder into the language model um as opposed to the vision encoder is a black box that just outputs a sequence of vectors.

Okay. So training um is uh so they have two phases. Um pre-training has actually four stages, post- training has three stages. So this you know you can see that this pipelines are getting quite uh complicated now. Um so they first train the adapter this is the same as all these models and then they have three stages where they progressively train on longer and longer sequences from 8k to 32k to 2 56 uh k. Uh most of the tokens are in the stages two and three. Um and and then for post- training they do SFT on long uh chain of thought data, knowledge distillation and then some reinforcement learning. Um um at this point it's really kind of a systems paper. There's you know there's a bunch of I highlighted the kind of the core new ideas but there's a lot of details that are are are different. And if you look at the final results the this is a pretty uh pretty good model. I mean if you look at these benchmarks as compared to the closed models Gemini um GPD5 and Opus 4.1 um you know these these numbers are um the bold means that it's the the best number in that in that row. So the quen models are actually quite quite strong.

Okay maybe the yeah question So, we're going to have Yeah.

>> Yeah. >> So the question is how do you if what about video generation and how do you know whether there it should be a video or text. So first of all these models don't generate video videos or images. The video all the multimodal stuff is on input side. you're always generating uh text and so um and then you're always in most of these stages except for RL you're supervising every token. So there's no you know uh LM as a judge or quality on how um well um the the description is working. It's just basically whatever is in your data set. Now with RL

you can play around with different you know reports.

>> Yeah. >> Yeah. Um I want to ask is the training of multiple model than training the pure LM from those system perspective perspective. for system. I think um the most um band would be used in uh images videos that test and from side I think uh image videos they have more to so that be >> yeah so the two questions are um are training multimodal models harder on the system side and I mean certainly it's not easier um um I think you pointed out that um the data sets are larger video data is just like you know even loading it can be bquares um generally we have not really focused on data loading for talking about language models because it's very very cheap so that needs to be um taken into account and you know all the same principles of making sure that your data loading is happening async with actual computation needs to be taken into account. And the second uh part was um sorry what was this second question?

>> They have more tokens. >> Oh yeah. So the videos and images can have more especially video have more tokens. So um that's one reason we saw this kind of normalization to to make sure that we're not um you know giving more weight to video. I mean you can always downweight. So if you have more tokens, if you don't like that, you can uh downweight. Um we talked to in the last or uh two weeks ago about like data mixtures. So you can always kind of weight things according to whatever um you know makes sense. Um but there's also a lot of text tokens out there. So I think in general um if you look at yeah the models are trained on like tens of trillions of tokens. Um, I think there's a Yeah, I wouldn't say that the number of multimodal tokens like vastly outnumbers text tokens.

>> Yeah. Oh, I think you were first. >> Yeah. >> Yeah. Yeah. So the question is um when you do this alignment um how does it know what the language is the model language model uh pre-trained and it's definitely has to be pre-trained because otherwise there's it doesn't make sense to align it. So the language model is frozen and you're just training the adapter to connect the the given uh vision encoder with the with a given uh language model.

And uh the way you train it is well you pick a token budget 67 billion um you know tokens and you just train. There's not a like adaptive threshold here. >> Yeah. also let me know about the difference between the brand number. So I notically this the size of the vision that So >> yeah. So the question is are um the number of parameters in the vision code are much uh smaller and the answer is uh yes in general. I'm trying to find um where is the other place I um not here. Okay. Um the reason is that the vision encoder is in some sense doing a very local you know operation. It's looking at a patch.

A patch is very small and it's just trying to understand the patches. There's not much knowledge. Um so to patch we're not reasoning. So most of the capabilities of the model are are still kind of in the language model. Um I'm trying to find the okay where maybe it was up here. Okay. I can't Oh yeah. Um so if you look at the lava model so the full so it's 72 billion parameter language model um and um show you what okay I don't know why this number is uh so oh this sorry this is 72 million okay so this is uh much smaller the projector is much smaller and the the vit is generally less than a billion you know uh parameters. Um yeah I guess mostly because it's fairly um kind of a local operation. I guess that's the easy way to say it. Okay, I'm going to think some other questions. I'm

going to try to move on now. Um so so I guess quen 3 is a kind of the the last vision language model I'll talk about. this got uh save our performance.

Um um there's a lot of you know data work that happens not too many details about the data and the data mix um in these later quen models. Um so you'll have to look at um the lava paper or the um the AI2 has a mobile paper which I didn't get a chance to talk about that has more details here. Um and you know from Quen to Quen to Quen 3 many of these changes were mostly um you know there's some kind of changes there um but the overall framework is remains the same and mostly it's kind of scaling up curating more data sets noticing that you know your long context isn't you have to handle long context and maybe sharpening your image processing and so on.

Okay. So, I'm going to now jump to um something quite different. So, there's this paper called Chameleon from Meta in 2024. Um so, so far VLMs encode images into these vectors and then they inject them into a language model. And because it's a language model, you can only generate text. you can't generate you know images. Um so you know there's many ways you can go about fixing this. You can just take a VLM. You can also attach a diffusion head and you now you can generate images or but I want to talk about this you know idea here which is chameleon says what if we mapped everything into discrete you know tokens and in some ways aesthetically this is uh well maybe this reflects that I'm a language person this is kind of appealing um um because now you can analyze and generate images in the same way everything is a discrete token If you want to do um you know uh you know what let's say you want to generate you know images uh you can say here is a here's a text prompt and then you can you know just you know these are more just tokens that you can generate or you can kind of uh flip flip uh the you know the language and the text around as well. um they have these examples um and from the command paper you can prompt um and saying I'm bored can you show me some birds and then it would be text and then there' be images and then some more text and some images so it's interled and you don't so the in some sense the vision of an omni model is that text and images truly live in the same space and this is accomplished by um making everything look like text that's one way to do it okay so I'll talk a a little bit about how this is done. So the thing um to you need to do now um is to map images into discrete tokens. So there's this idea rather old uh from um uh from or 2017 which is called vector quantized variation encoder and the basic idea here is to uh map an image into um a discrete code and what you learn is a mapping that maps this into a continuous uh vector and this gets sort of rounded to the nearest code. So you have a code book which has let's say 8,000 uh codes.

So these are like prototypical uh vectors that correspond to what you know patches and then so you pick uh you know one of these that's the closest and that would be your representation. So that's your encoder. It goes through a continuous phase and then you sort of round it to the nearest code. And now the decoder is you take that code and you try to reconstruct the image and then you train these VQEs by um you know uh basically minimizing the reconstruction loss. Um there's some other terms that you add because this is not differentiable um which I won't have time to get into.

Okay, but at the end of the day, you have 512 x 512 images that are converted into a,024 tokens. Um, and each token is kind comes from a vocabulary of 8,000. Okay. So now you have basically text, you have your images,

um, which are basically these codes. um they train a new tokenizer uh to you know now because now you have different sort of your data looks different than if it were just um normal uh natural language. Um the training is is um now actually very straightforward. This is just normal language model training, right? There's no adapter, there's no um you know turn the vision encoder or there's there's just a language model now. So this is the part that's kind of appealing. Um so there's still two stages but you know even for language model training there's usually two or more stages where the first stage is the bulk of training is large unsupervised um they have you know both text and text and image um and then stage two they mix in high quality data.

Okay, so this is really great and and elegant. Um there's a few problems with this. One is that um they found that the training was not stable. And the reason is that you know text and images despite occupying the same space just behave very differently. So just calling things discrete tokens isn't really, you know, isn't hiding the fact that there's an image living there. And in particular, text tokens, if you think about predicting the next word, this has relatively lower entropy. Most words are kind of predictable, whereas image tokens um have very high entropy.

I don't know what shade of blue this exact token is going to be. Um and so as a result they found that this training with this kind of mixture led to a lot of the norms of um the parameters to grow um and to generate kind of loss um instability. They were able to kind of mitigate or fix this to some extent via via QK norm and Z loss regularization which controls the norm growth. Um so so um I'm not even going to uh show the results but um I wanted to highlight this work because there's a certain elegance here because it's just one model that treats really all modalities the same. But the downside is that um it turned out this model was not really as performant and the discretization does definitely loses information. Um so think about uh you know OCR um you know again if you discretise um you know very small print you're not going to be able to read it anymore. And finally training with multimodalities is also tricky. We kind of saw this a little bit with the quen models and we had to you know play around with the weights but this is more exacerbated here. So you know VKU VAEs were kind of popular for a while and then and in fact you know many people were using this for image generation. The main reason to do that is well you have a transformer um how do you generate from a transformer?

You well you have to generate discrete things and you basically put all your you know data into this discrete form. But then you know diffusion models uh came out um and became popular and viable for generation. So this flavor of method is sort of uh less uh less popular than it used to be. Okay. So um maybe just kind of reflect and summarize here. Um you know frontier models these days are expected to be multimodal or even more strongly natively multimodal or omnimodels. Um, you know, unfortunately there's, you know, I think when Gemini comes out or GPD comes out, they tout are touted as being natively multimodal and handle all these modalities and in fact they do.

Um, but of course there's no details about how these are are built. Um I think that you know it's probably some you know combination of uh having a continuous encoder because you don't want to lose information and then diffusion for the generation because but that's you know my speculation. Um you know the fundamental challenge I think when you're dealing with multimodality is how do you handle non-ext modalities? Um it's also interesting that there's a symmetry between you know understanding the modality and generation and there's

different there's no one universal encoder let's say for example in when we looked at clip um you only cared for classification to capture high level semantics so these vectors could be fairly small and capture the high level semantics whereas if you wanted to do OCR or whether you wanted to generate an image then you need really fine grain detail like you need to um that's why diffusion is so good because it can like really micro um optimize the the the low high frequency information um you know I think in general when you're dealing with multimodalities uh you have to think carefully about um weighing them properly um for example video certainly has lower information density than text um so you don't want a video to overwhelm your your your text. Um and like I said before um perhaps the at least the current best um thing is having continuous encoders. You know even you know it seems like even clip even though it's five years old or similar ideas are still kind of the go-to way to capture semantics of of of images. Transformers are still there. Um and then uh diffusion models which I didn't talk about are are great for you know generation.

Okay so I will stop there. Um that's uh it for um the lecture on multi modality. Um I guess we don't have any homework on uh doing this but if you're curious um it would be I think I encourage you to play around with um you know training some of these models.