# Task description

תיאור מקיף של מה רואים בתמונה. צריך לעבוד על סוגי התמונות הבאות:

1. צילומי מסך (יותר impact)
2. תמונות מהמצלמה הראשית של הטלפון

# Types of tasks

1. Image Captioning - <https://paperswithcode.com/task/image-captioning>
   1. + Text Retrieval on captions ?
2. Image-Text Retrieval - <https://paperswithcode.com/task/image-text-matching>
   1. Image-text matching is one of the fundamental topics in this field, which refers to measuring the visual- semantic similarity between a sentence and an image. It has been widely adopted to various applications such as the retrieval of text descriptions from image queries or image search for given sentences.
3. Document understanding - <https://paperswithcode.com/task/document-understanding>
4. Document Visual Question Answering
   1. For each user query prepare a question and run the model on all images
   2. Pros: Full flexibility
   3. Cons: Expensive and actually infeasible
5. Classification / Multi-label classification (maybe via object detection)
   1. Pros: Can run on all images in advance and put in database
   2. Cons: Need to decide on labels in advance, not extracting the full data (for example text in browser)

# Models

## General Models

1. GIT
   1. By Microsoft
   2. Image-Encoder (pretrained on contrastive Task) + Language Decoder
   3. Add the image output tokens at beginning of decoder task
   4. Support Image Captioning
   5. Git uses CLIP do\_center\_crop of 224\*224 (<https://huggingface.co/transformers/v4.6.0/_modules/transformers/models/clip/feature_extraction_clip.html>) - maybe it’s better to rescale ourselves
   6. The original git model is not open source. Only two smaller variants: git-base and git-large
      1. <https://huggingface.co/datasets/naorm/desktop-git>
      2. <https://huggingface.co/datasets/naorm/desktop-git-large> - Better than git-base. Give complete sentences with more context
      3. <https://huggingface.co/datasets/naorm/desktop-git-large-textcaps> - Focuses on the text/words in the screen but loses the overall context. Worse overall but can be good complimentary
   7. Much slower than blip (1.5-3x slower) - 13s vs 4-8 s
2. BLIP
   1. By Salesforce
   2. Complicated architecture that support cross attention between Image Encoder and Text Decoder and train for both contrastive and generative loss
   3. Support Image Captioning + Image-Text Retrieval
   4. There are two distributions of the model
      1. Huggingface
      2. LAVIS (salesforce library) - By benchmarks and eye test seems better on average that Huggingface version but saw some observed cases Huggingface outperformed (in identifying calendar and video editing tool)
   5. They are two sizes of the model
      1. Base - 2.2s
      2. Large - 4.4s
3. BLIP-2
   1. Depend on two frozen pre-trained image encoder and language model (decoder-only or encoder-decoder) and QFormer transformer layer that is trained to bridge between the two
   2. The smallest model is with 2.7B parameters. Takes 10s for single inference on Google Colab and I wasn’t able to load it to memory on my docker

## Document Models

1. Document models should be better at reading texts and understanding how they fit with the image
2. DiT - Document Image Transformer (by microsoft)
   1. <https://huggingface.co/microsoft/dit-base-finetuned-rvlcdip>
   2. <https://arxiv.org/abs/2203.02378>
3. Donut - Document Understanding model
   1. Swin Transformer + BART
   2. Tasks
      1. Classification
      2. Document Information Extraction
      3. Document Visual Question Answering
   3. Available models
      1. <https://huggingface.co/naver-clova-ix/donut-base>
      2. <https://huggingface.co/jinhybr/OCR-Donut-CORD>
4. Pix2Struct
   1. Image-encoder - test-decoder based on ViT
   2. Pre-trained to predict HTML source based on screenshot
   3. Keeps the aspect ratio
   4. Available models
      1. <https://huggingface.co/google/pix2struct-base>
      2. <https://huggingface.co/google/pix2struct-ai2d-large>

## Instruction-based Models

elaborate more but more hallucinate

1. Llava - <https://arxiv.org/abs/2304.08485>
2. InstructBlip - <https://arxiv.org/abs/2305.06500>

# General Model Comparison

Aggregated results

* <https://huggingface.co/datasets/naorm/webscreen-test-captions>
* <https://huggingface.co/datasets/naorm/desktop-ui-captions>

All performance results are on Docker (Docker is 2-3x faster than jupyter lab - not sure why since both are using the same underlying host - i.e my computer)

## Website-screenshots

| Model | Optimization | min\_length |  |
| --- | --- | --- | --- |
| git-large |  |  | avg\_caption\_size=7.553719008264463, avg\_time=**9.123038867288384**, std\_time=3.565055158829227, median\_time=8.381808400154114, 95pct\_time=14.414076948165889 |
| blip-large |  |  | avg\_caption\_size=13.479338842975206, avg\_time=**4.48863100808514**, std\_time=1.6603041985573537, median\_time=4.241687536239624, 95pct\_time=5.9281683325767505 |
| blip-caption/base |  | 10 | avg\_caption\_size=10.107438016528926, avg\_time=**2.297469253382407**, std\_time=0.4177590236518318, median\_time=2.21173357963562, 95pct\_time=3.097647702693939  \_\_\_\_  Torch.compile even makes it worse:  avg\_caption\_size=10.107438016528926, avg\_time=2.803495661286283, std\_time=0.5564380934789553, median\_time=2.7626630067825317, 95pct\_time=3.823147356510162 |
| blip-caption/large |  | 10 | avg\_caption\_size=10.041322314049587, avg\_time=**3.9480137539303994**, std\_time=0.40599535799759756, median\_time=3.8702497482299805, 95pct\_time=4.645302045345306 |
| blip-large-5beam |  |  | avg\_caption\_size=13.297520661157025, avg\_time=**15.082509862489937**, std\_time=9.823130888139703, median\_time=12.897488951683044, 95pct\_time=22.0889808177948 |
| **Min 20 tokens** | | | |
| blip-large |  | 20 | avg\_caption\_size=22.15702479338843, avg\_time=**5.257788860108241**, std\_time=0.8223930726206491, median\_time=5.0598297119140625, 95pct\_time=6.206869852542876 |
| blip-caption/base |  | 20 | avg\_caption\_size=19.09090909090909, avg\_time=**3.382509276886617**, std\_time=1.1484465113547806, median\_time=3.089022994041443, 95pct\_time=4.99631671905517 |
| blip-caption/large |  | 20 | avg\_caption\_size=13.914715297052553, avg\_time=**9.895143467532701**, std\_time=44.56820452113893, median\_time=5.3237515687942505, 95pct\_time=6.623675918579101 |
| Pix2struct-base | torch.compile |  | avg\_caption\_size=6.479338842975206, avg\_time=11.51344880683363, std\_time=3.1369140505383806, median\_time=10.831488013267517, 95pct\_time=16.164117014408113 |
| Pix2struct-large | torch.compile |  | avg\_caption\_size=5.966942148760331, avg\_time=39.51728412730635, std\_time=3.702268907207008, median\_time=39.803780913352966, 95pct\_time=44.258911192417145 |

## Desktop-ui

| Model | Optimization | min\_length |  | RAM size |
| --- | --- | --- | --- | --- |
| blip-large |  |  | avg\_caption\_size=12.764705882352942, avg\_time=**5.0537688124413584**, std\_time=1.237462424665482, median\_time=4.761080026626587, 95pct\_time=6.630144000053406 |  |
| blip-caption/base |  | 10 | avg\_caption\_size=10.92156862745098, avg\_time=**2.375835479474535**, std\_time=0.4516113502276029, median\_time=2.3237459659576416, 95pct\_time=2.9541900157928467  \_\_  Torch.compile makes it worse:  avg\_caption\_size=10.92156862745098, avg\_time=2.868942592658249, std\_time=0.47169184951598536, median\_time=2.8068389892578125, 95pct\_time=3.6484330892562866 |  |
| blip-caption/large |  | 10 | avg\_caption\_size=10.823529411764707, avg\_time=**4.0545122810438565**, std\_time=0.693701273925722, median\_time=3.8660941123962402, 95pct\_time=5.484575867652893 |  |
| git-base | None |  | avg\_caption\_size=7.784313725490196, avg\_time=3.55586249220605, std\_time=1.4203826457443074, median\_time=3.202315092086792, 95pct\_time=6.291345000267029 |  |
| git-large |  |  | avg\_caption\_size=9.254901960784315, avg\_time=**16.386740170273125**, std\_time=11.46767632695501, median\_time=12.919917106628418, 95pct\_time=40.27289593219757 |  |
| git-base-coco | torch.compile |  | avg\_caption\_size=8.333333333333334, avg\_time=3.5879543435339833, std\_time=1.016185660691769, median\_time=3.370856761932373, 95pct\_time=5.419468998908997 |  |
| git-large-coco | torch.compile |  | avg\_caption\_size=10.156862745098039, avg\_time=15.131292474036123, std\_time=4.3333391027397195, median\_time=13.948282718658447, 95pct\_time=22.639429330825806 |  |
| pix2struct-base | None |  | avg\_caption\_size=6.745098039215686, avg\_time=11.504128596361946, std\_time=1.228750077098563, median\_time=11.442872762680054, 95pct\_time=13.428338408470154 | 2-3GB |
| pixstruct-base | onnx |  | avg\_caption\_size=6.745098039215686, avg\_time=6.84844196543974, std\_time=0.6329323053603351, median\_time=6.619127035140991, 95pct\_time=7.73145854473114 |  |
| pix2struct-base | torch.compile |  | **Torch.compile makes worse**  avg\_caption\_size=6.745098039215686, avg\_time=11.591954595902386, std\_time=2.3485898057132655, median\_time=11.17323899269104, 95pct\_time=13.053064942359924 |  |
| pix2struct-base | triton |  | avg\_caption\_size=6.745098039215686, avg\_time=18.3355692134184, std\_time=4.535828714721957, median\_time=17.548560857772827, 95pct\_time=23.129679918289185 |  |
| pix2struct-large | None |  | avg\_caption\_size=6.333333333333333, avg\_time=49.46783588446823, std\_time=11.199333408125947, median\_time=44.74034905433655, 95pct\_time=71.28406095504761 | 7-8GB |
| pix2struct-large | torch.compile |  | avg\_caption\_size=6.333333333333333, avg\_time=41.58534959250805, std\_time=7.248787668075207, median\_time=40.407142877578735, 95pct\_time=46.781251549720764 |  |
| **Min 20 tokens** | | | |  |
| blip-large |  | 20 | avg\_caption\_size=22.80392156862745, avg\_time=**7.687416581546559**, std\_time=5.423281098136076, median\_time=6.731517791748047, 95pct\_time=9.83824598789215 |  |
| blip-caption/base |  | 20 | avg\_caption\_size=18.49019607843137, avg\_time=**3.475175796770582**, std\_time=0.716689601335981, median\_time=3.280345916748047, 95pct\_time=4.472548604011536 |  |
| blip-caption/large |  | 20 | avg\_caption\_size=19.137254901960784, avg\_time=**5.143004006030512**, std\_time=0.48538127362382555, median\_time=4.973120927810669, 95pct\_time=6.093285441398621 |  |

**BLEU scores**

BLIP score: 9.461496220616453

BLIP/20 score: 5.749816875690565

BLIP-caption/Base score: 12.719065773932035

BLIP-caption/Base-20 score: 8.804694433766668

GIT score: 4.113388278878379

BLIP-caption/Large score: 12.274092012232247

BLIP-caption/Large-20 score: 8.294888776316835

**ROUGE scores**

GIT score: {'rouge1': 0.32632603950704564, 'rouge2': 0.10998672485626904, 'rougeL': 0.3024706292525361, 'rougeLsum': 0.3023702089102898}

BLIP score: {'rouge1': 0.42060106190278296, 'rouge2': 0.1694946594101338, 'rougeL': 0.3590442906917677, 'rougeLsum': 0.35962354463269647}

BLIP/20 score: {'rouge1': 0.35521609076085137, 'rouge2': 0.1388985901229845, 'rougeL': 0.2958732674589946, 'rougeLsum': 0.2964996880795625}

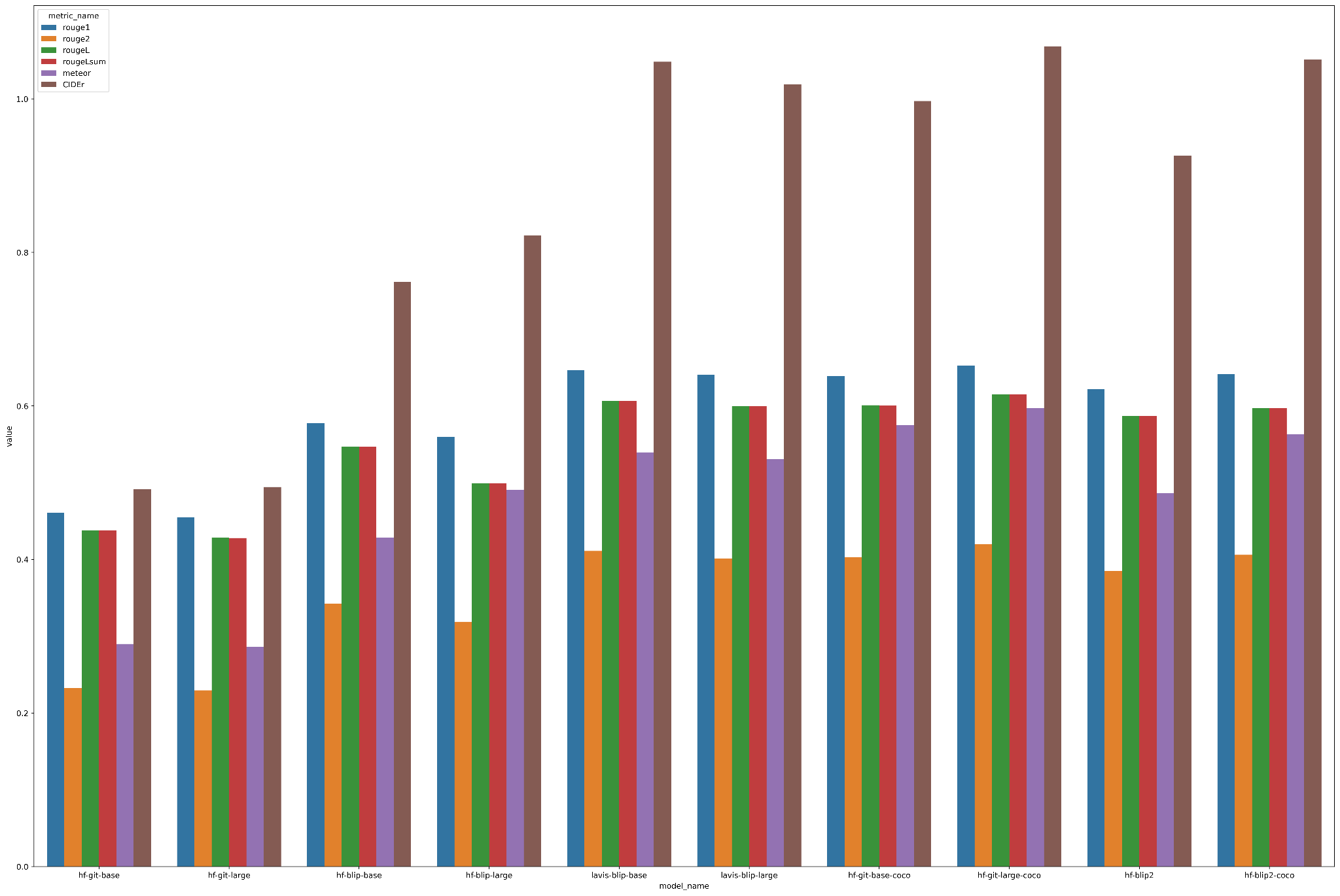
BLIP-caption/Base score: {'rouge1': 0.48209646304028325, 'rouge2': 0.2270207461515743, 'rougeL': 0.43458663491209104, 'rougeLsum': 0.43538516877731076}

BLIP-caption/Base-20 score: {'rouge1': 0.4065971051745647, 'rouge2': 0.1824901993527107, 'rougeL': 0.3581547990548702, 'rougeLsum': 0.35878621331090077}

BLIP-caption/Large score: ROUGE score: {'rouge1': 0.48418999296579307, 'rouge2': 0.22332857950068538, 'rougeL': 0.4376091123409079, 'rougeLsum': 0.43682785306495886}

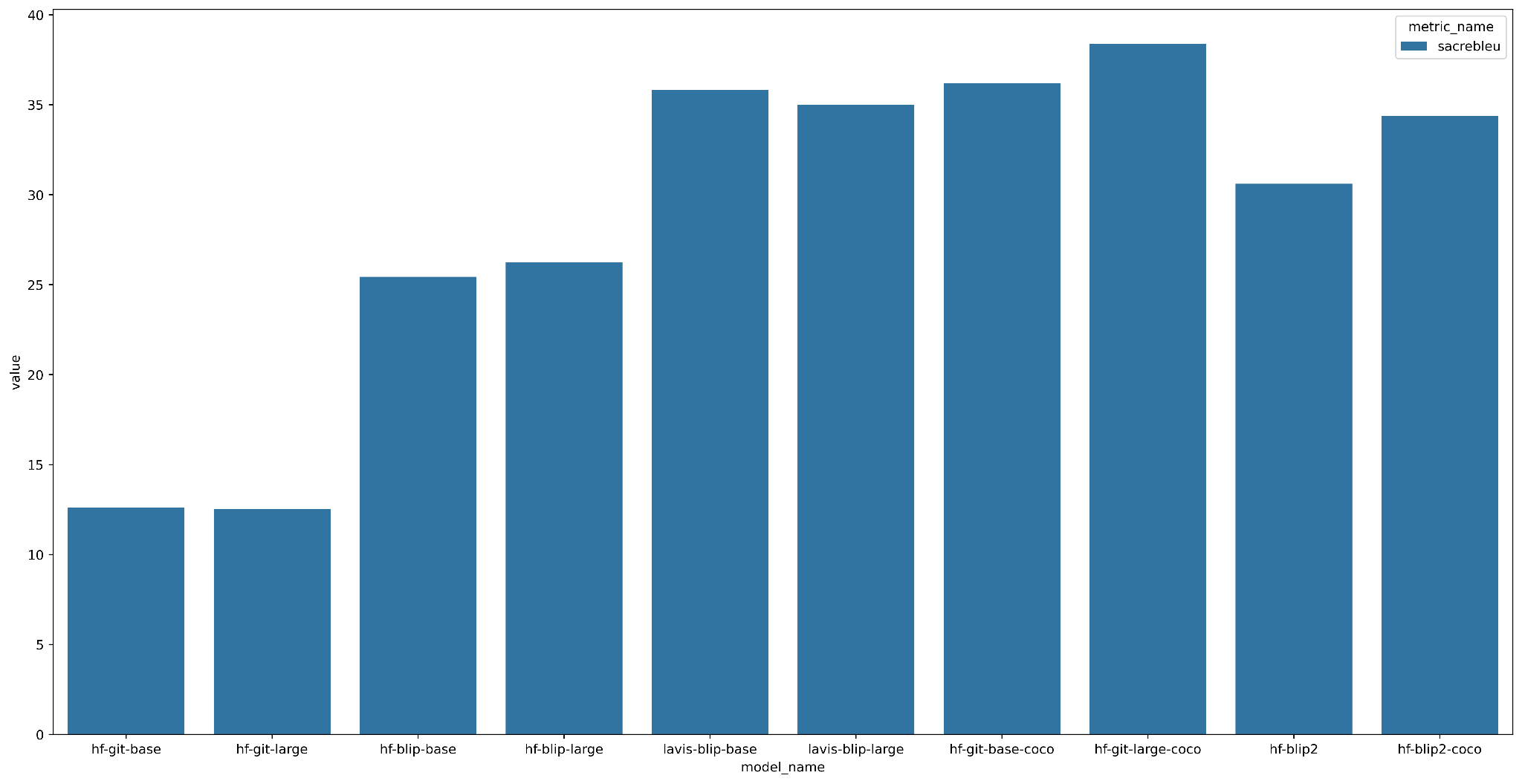
BLIP-caption/Large-20 score: {'rouge1': 0.3995055155231303, 'rouge2': 0.17311055122928592, 'rougeL': 0.35380241500846366, 'rougeLsum': 0.3537090822372009}

## Coco Captions Evaluation

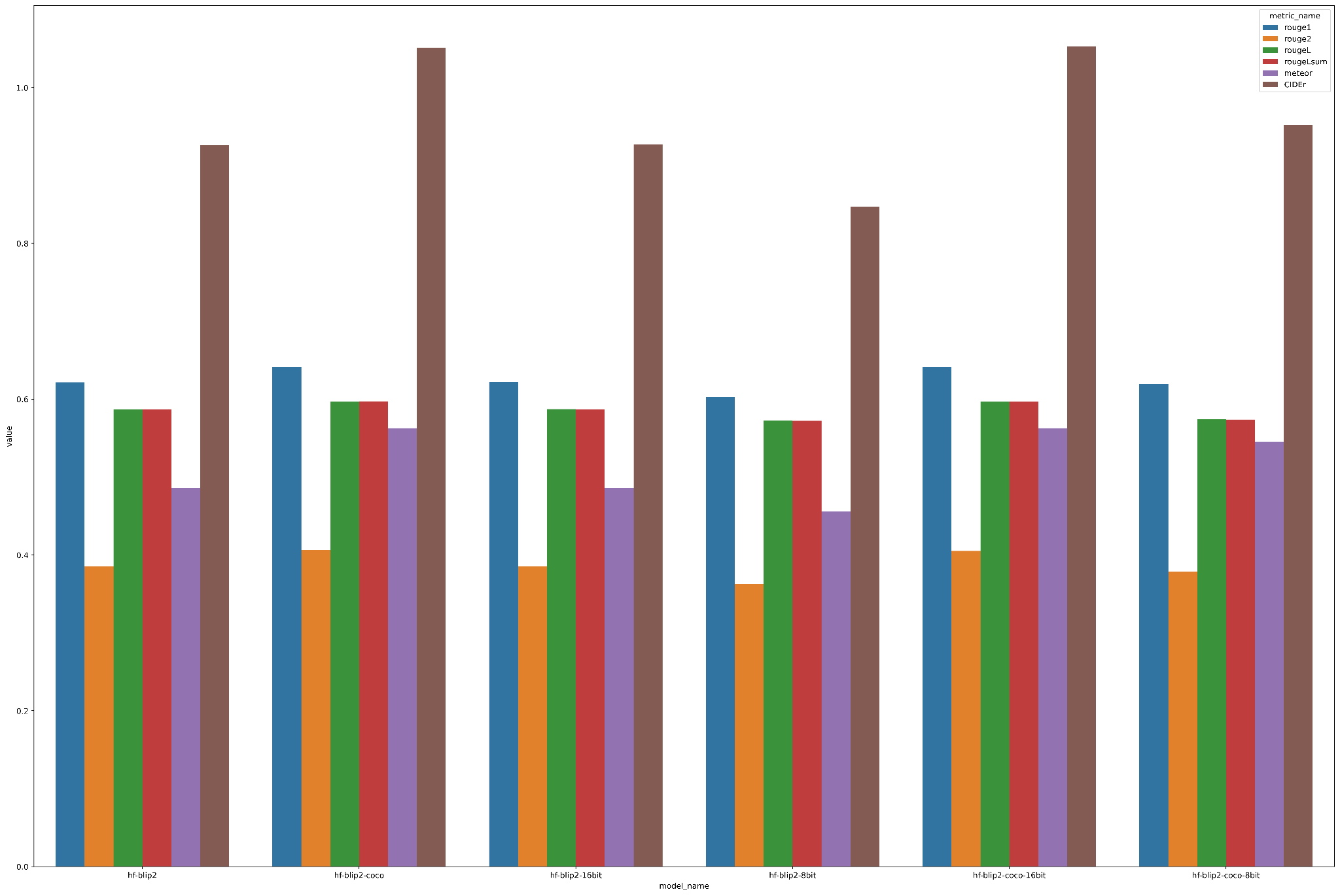


Conclusions of chart:

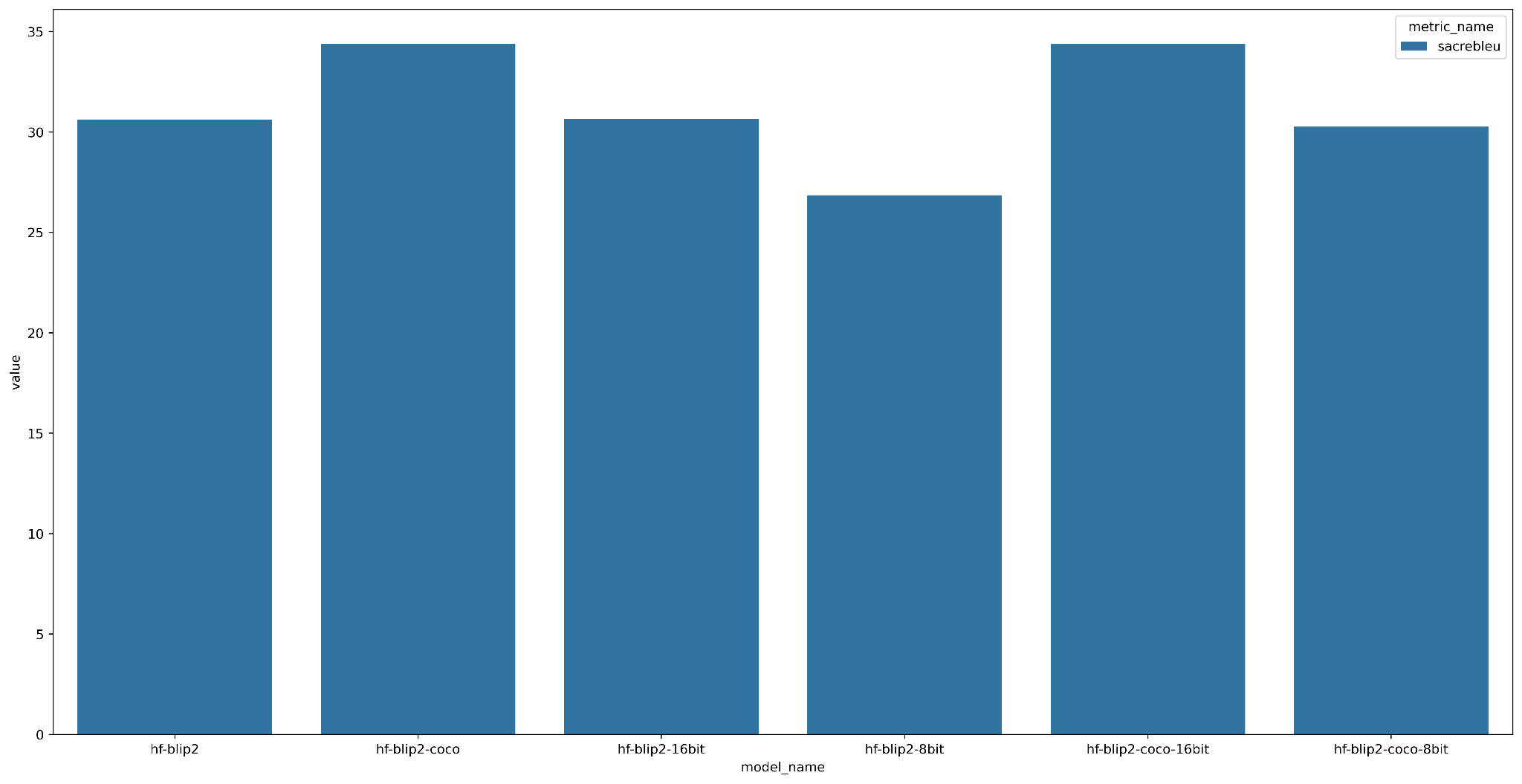
1. hf-git-base/large are the worst performing
2. Lavis-blip are way better than hf-blip
3. The best performing is git finetuned on coco
   1. A bit behind is hf-blip2-coco that could be faster
   2. After them is lavis-blip-base which should be more faster
4. Surprisingly lavis-blip-base perform better than lavis-blip-large



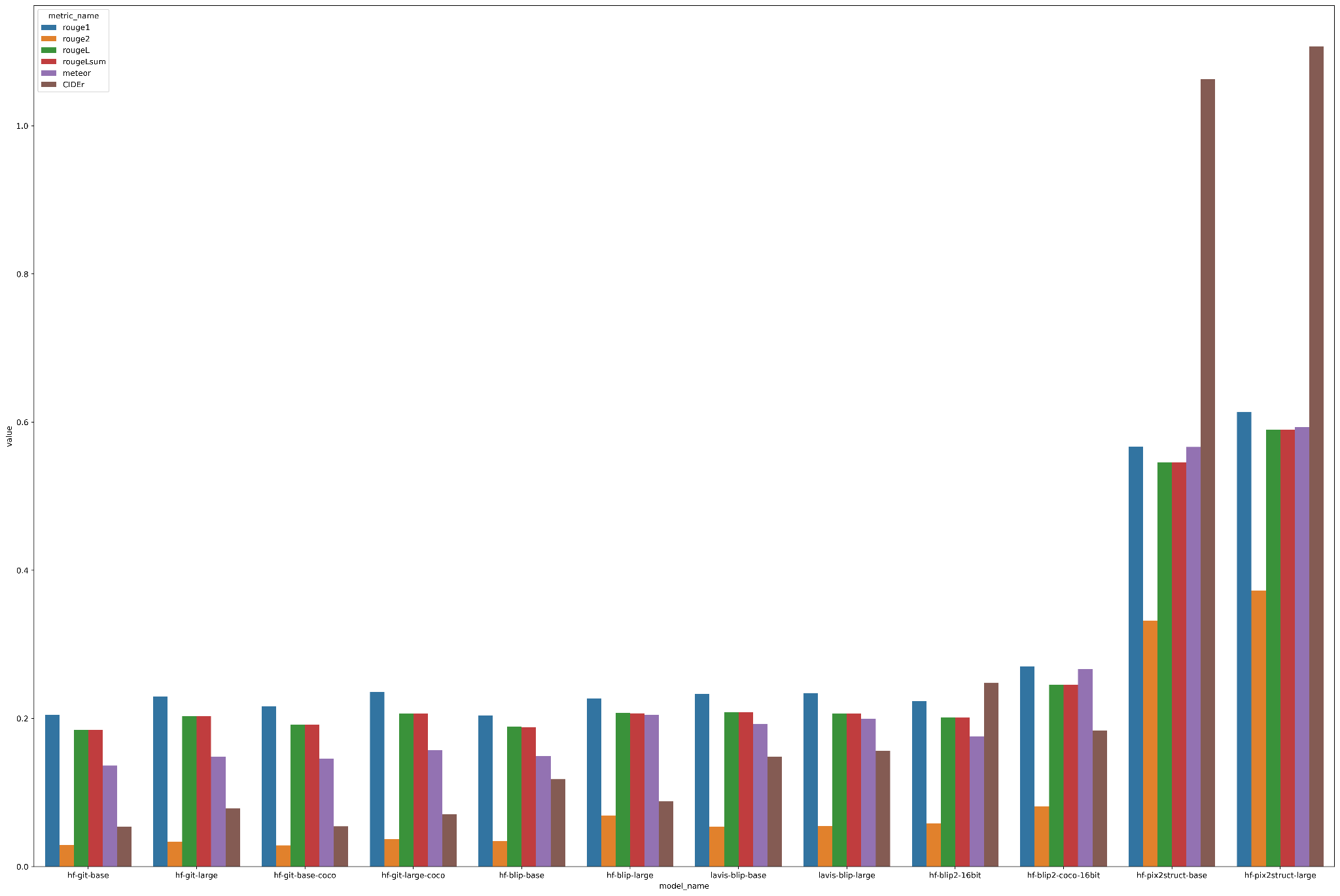
With scarce blue it actually looks like lavis-blip is better and hf-git are better than hf-blip2

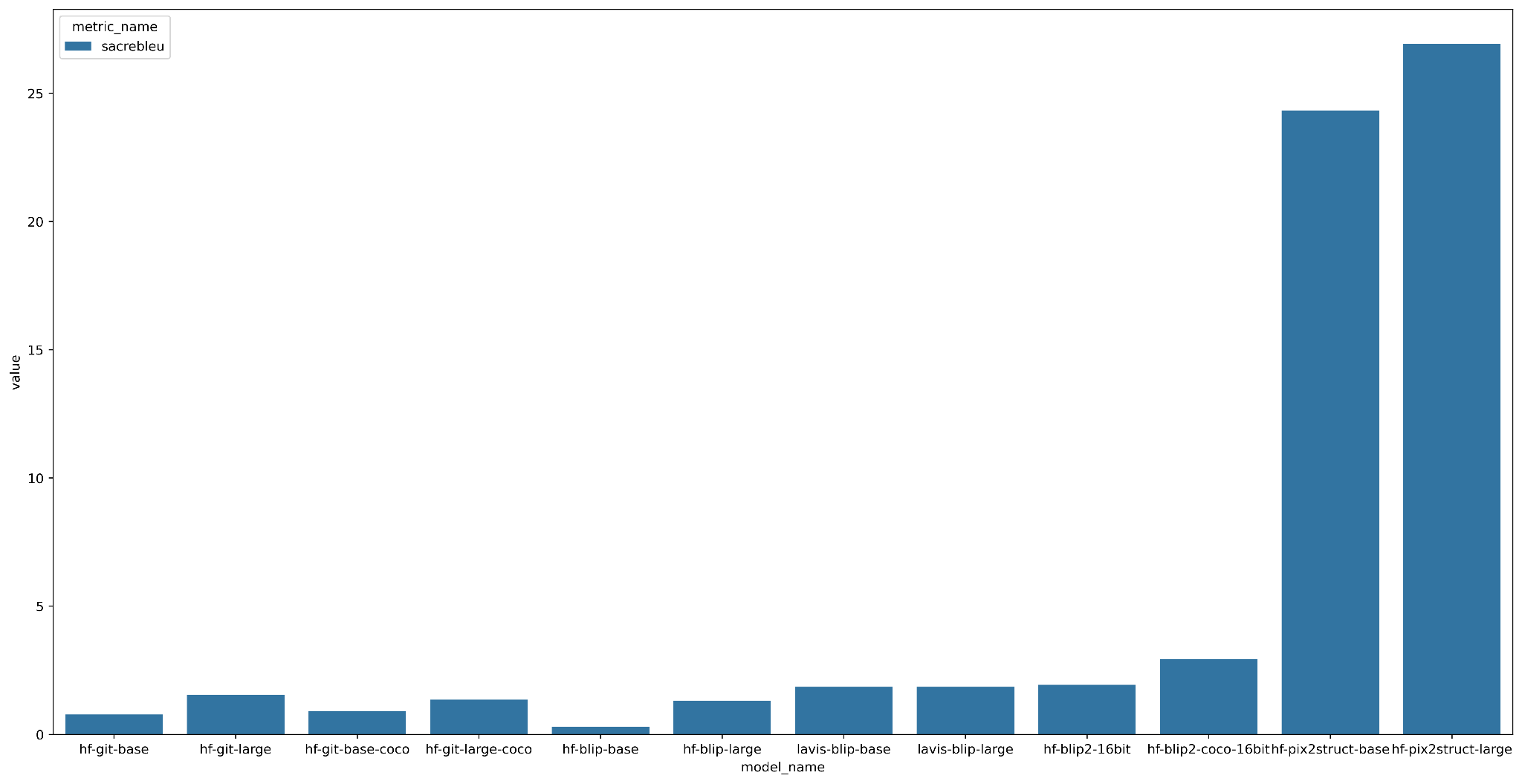


From this chart we can see that 16bit version has almost the same performance like regular version but there is dropoff with 8bit



## Screen2words Evaluation



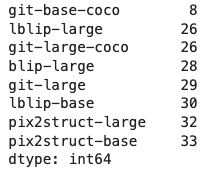


## Performance on Desktop-UI

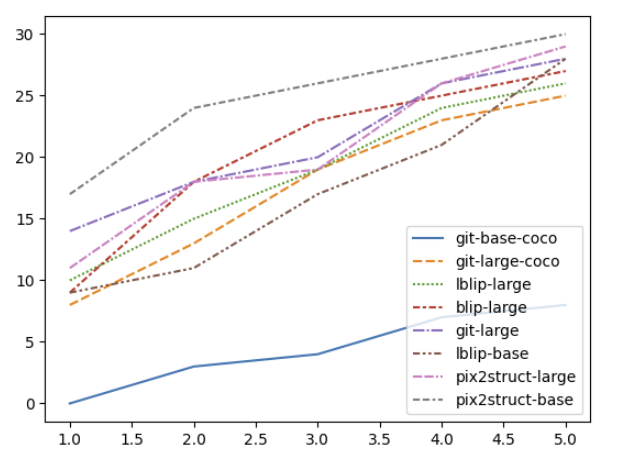
# 

## Desktop-UI Evaluation

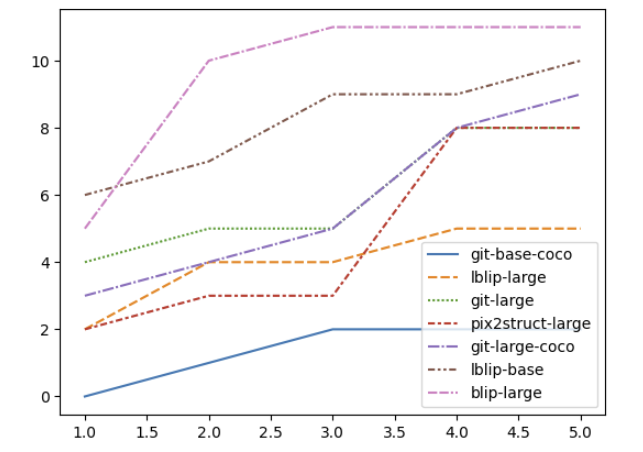
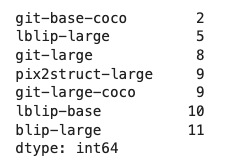
Pix2struct-base has the most correct captions:

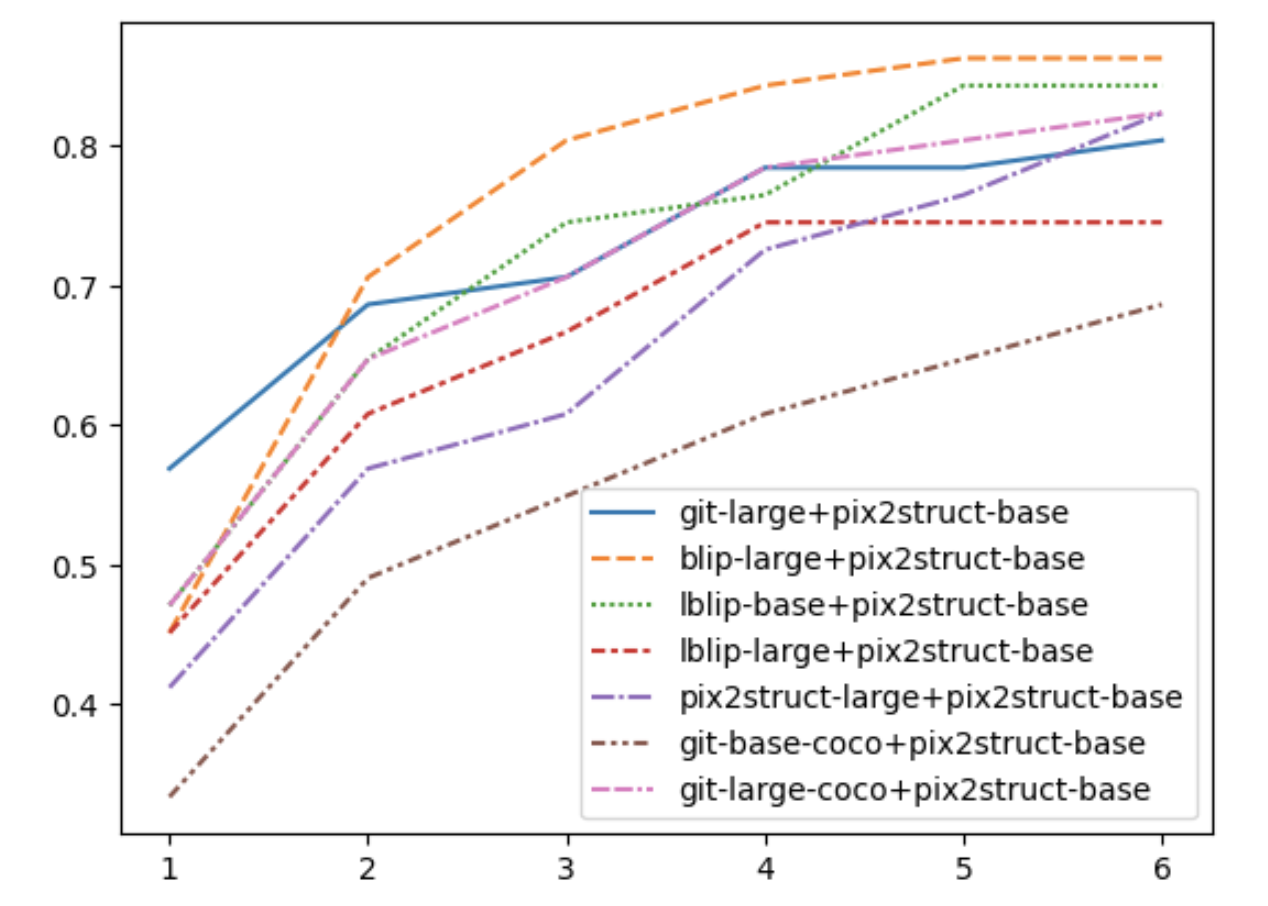


Also if you look on captions of what model are at the top-1, top-2, top-3, top-4, top-5, Pix2struct-base also wins



Another thing that I looked at was what models complement pix2struct-base in scenarios where it failed to product correct caption





Blip-large is best but lblip-base is second best and it’s also faster so it’s probably best to go with it

**Conclusion:** For desktop-ui run pix2struct-base and you can complement it well with lblip-base

# Datasets

## General Images

1. COCO captions - <https://cocodataset.org/#download> (for image captioning)
2. Flickr 8K
3. Flickr 30K
   1. For Image-Text Retrieval <https://paperswithcode.com/sota/image-to-text-retrieval-on-flickr30k>
4. TextCaps - <https://textvqa.org/textcaps/>
5. Visual Genome
6. NoCaps

## Document Images

For reading documents/OCR:

* <https://huggingface.co/datasets/RIPS-Goog-23/IIT-CDIP> - set of 11M scanned english document images

For captioning:

* **Screen2Words**
  + Papers using it - <https://paperswithcode.com/dataset/screen2words>

For classification:

* **RVL-CDIP** - 400K images in 16 classes, with 25K images per class. The classes include letter, memo, email, and so on. There are 320K training, 40K validation, and 40K test images.

For Document Information Extraction:

* **CORD**. The Consolidated Receipt Dataset (CORD)3[45] is a public benchmark that consists of 0.8K train, 0.1K valid, 0.1K test receipt images
* **Ticket**. This is a public benchmark dataset [12] that consists of 1.5K train and 0.4K test Chinese train ticket images

For Document Visual Question Answering:

* **DocVQA** - The dataset is from Document Visual Question Answering competition4 and consists of 50K questions defined on more than 12K documents

## Screenshots

1. <https://github.com/waltteri/desktop-ui-dataset> - 51 images fully annotated of desktop screenshots
2. <https://public.roboflow.com/object-detection/website-screenshots> - annotated website screenshots

# Tasks

1. Finish reading about GIT and relevant blog posts
2. Try out basic captioning models on natural images
3. Try out basic captioning models on desktop datasets
4. Upload results to HuggingFace for easy view
5. Try out another <https://public.roboflow.com/object-detection/website-screenshots> dataset
6. Deploy both models in docker (get in bulk)
   1. Start with images
   2. Then add the rest for screenshots
7. Check decoding strategies (need to check how much it improves and also how much it slows down inference)
8. Compare between GIT and BLIP in desktop-ui
9. Contact Liron about the times to decide if to optimize
10. Compare between GIT and BLIP in standard benchmarks - coco captions (with evaluation)
11. Use BLIP-2 for captioning, check performance and efficiency
12. Create two dataset (desktop-ui and screenshots) with image and all available captions
13. Change min\_length in each of the models
14. Replace SBERT model with the asymmetric model
15. Try out image clustering
16. Arrange all documentation about this task
17. Add ONNX implementation to Blip to improve its speed/size
18. Try out nucleus sampling (beam search was too slow) <https://towardsdatascience.com/decoding-strategies-that-you-need-to-know-for-response-generation-ba95ee0faadc>
19. Create metric that only extracts ROUGE recall and not f1 to better measure the contribution of increasing min\_length
20. Try to run BetterTransformer on HF Blip2
21. Try out document captioning models

# Study Plan

1. ViT paper (An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale)
2. Pix2Struct: Screenshot Parsing as Pretraining for Visual Language Understanding
3. You Only Look Once: Unified, Real-Time Object Detection
   1. <https://docs.ultralytics.com/#yolo-a-brief-history>
   2. <https://sidharkal.medium.com/image-classification-with-yolov8-40a14fe8e4bc>
4. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation
5. GIT: A Generative Image-to-text Transformer for Vision and Language
6. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows
7. Florence: A New Foundation Model for Computer Vision
8. Cvt: Introducing convolutions to vision transformers
9. Unified Contrastive Learning in Image-Text-Label Space
10. Read relevant blog posts
    1. Read <https://asiffer.github.io/posts/desktop-elements-detection-using-deep-learning/>
       1. Github: <https://gitlab.com/d3sker/desker>
    2. Read <https://ankur3107.github.io/blogs/the-illustrated-image-captioning-using-transformers/>
11. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models
12. Dense Text Retrieval based on Pretrained Language Models: A Survey
13. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks
14. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (T5)
15. Sentence-T5: Scalable Sentence Encoders from Pre-trained Text-to-Text Models
16. Learning Transferable Visual Models From Natural Language Supervision (CLIP)
17. TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models (to understand VisionEncoderDecoderModel)
18. DocLLM: A layout-aware generative language model for multimodal document understanding
19. Document AI
    1. <https://paperswithcode.com/paper/markuplm-pre-training-of-text-and-markup>
    2. <https://paperswithcode.com/paper/blip-adapter-parameter-efficient-transfer>

# Future Deployment Optimization

<https://github.com/huggingface/text-generation-inference>