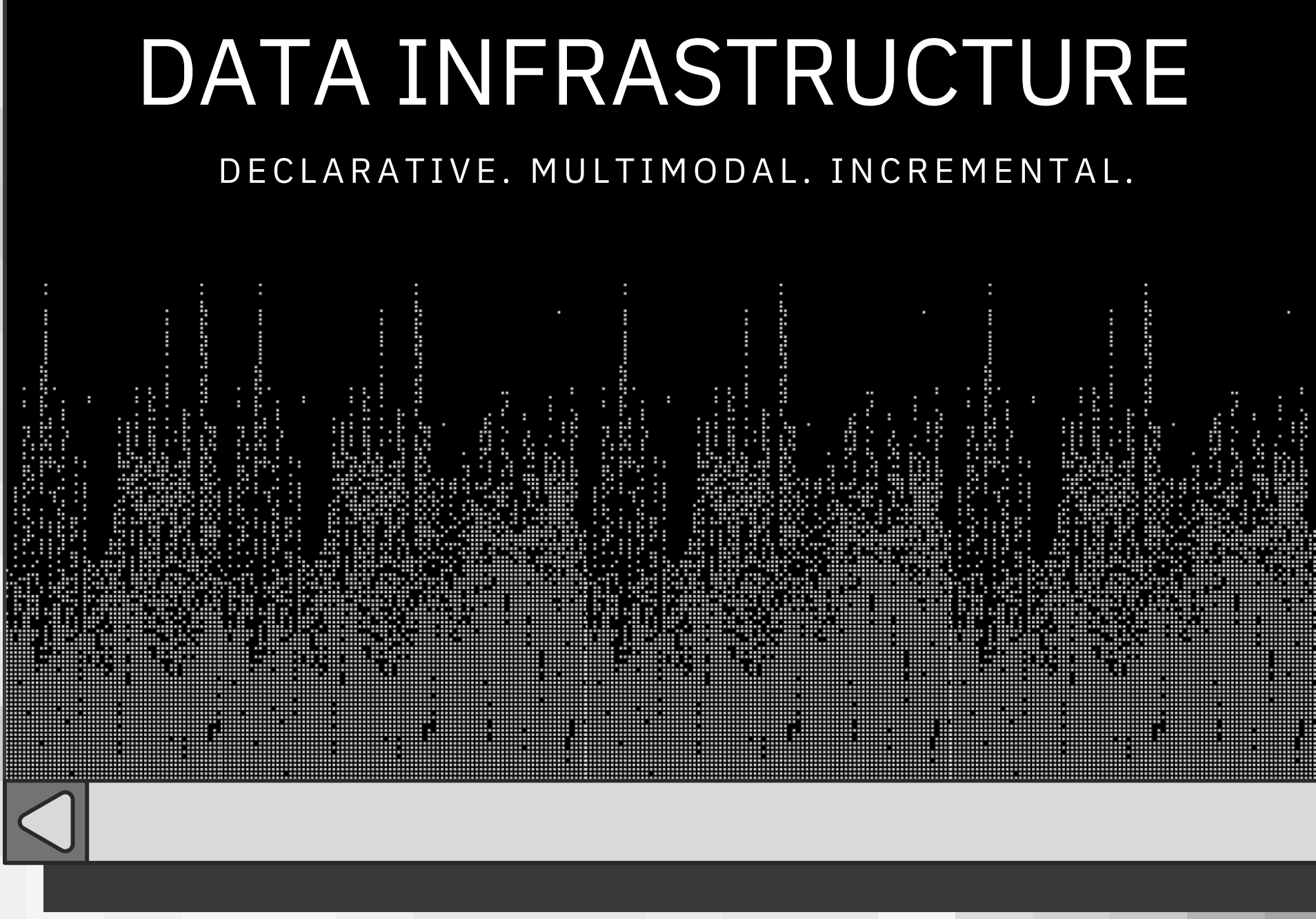
Pixeltable

OPEN SOURCE AI DATA INFRASTRUCTURE

DECLARATIVE. MULTIMODAL. INCREMENTAL.



Declarative

Your AI workflows: Data, Model, and Inference

```
# Persistent and versioned table holding data
video_table = pxt.create_table('videos', {'video': pxt.Video})

# Insert URL, Local Files, CSV, Blob Storage, Pandas, HTML, PDF...
video_table.insert(video='https://..../video.mp4')

# Automate data operations with views and build custom iterators
frames_view = pxt.create_view(
    'frames',
    video_table,
    iterator=FrameIterator.create(video=video_table.video))

# Co-locate one or many embedding indices
frames.add_embedding_index(‘frame’,
string_embed=clip_text.using(model_id='openai/clip-vit-base-patch32'),
image_embed=clip_image.using(model_id='openai/clip-vit-base-patch32'))

# Flexible query language even for similarity search
sample_text = 'red truck'
sim = frames_view.frame.similarity(sample_text)
frames_view.order_by(sim, asc=False).limit(5).select(
frames_view.frame, sim=sim).collect()
```

AI DATA INFRASTRUCTURE — DECLARATIVE, MULTIMODAL, INCREMENTAL

Multimodal

Easily work with Array, images, JSON, video, audio & more...

```
# Computer Vision model invocation
detr_for_object_detection(table.column,
model_id='facebook/detr-resnet-50', threshold=0.8)

# Extract audio from videos
calls.add_computed_column(audio=extract_audio(
    calls_table.video, format='mp3'))

# Work with metadata
calls.add_computed_column(metadata=get_metadata(calls_table.audio))

# From audio to text
calls.add_computed_column(transcription=openai.transcriptions(
    audio=calls_table.audio,
    model='whisper-1'))

# LLM model invocation
calls.add_computed_column(insights_response=mistral.chat_completions(
    messages=calls_table.insights_prompt,
    model='open-mistral-nemo'))

# Bring Your Own Code
@pxt.udf
def get_sentiment_score(text: str) -> float:
    return TextBlob(text).sentiment.polarity
```

Incremental

When new data is inserted, Pixeltable automatically...

```
# Only processes the new row, not the entire dataset
chat_table.insert([{
                    'user_id': user_id,
                    'question': message.content,
                    'timestamp': datetime.now()
                }])

# Updates any computed columns that depend on the new data
# Refreshes relevant indices incrementally
# Maintains all data relationships and lineage
result = chat_table.select(
            chat_table.response).order_by(chat_table.timestamp,
            asc=False).limit(1).collect()

response = result['response'][0]
```

Use Cases

Build Production-Ready AI Applications with Pixeltable

```
# Data Store and Computer Vision Workloads
frames_view.add_computed_column(detect_yolox_tiny=yolox(
          frames_view.frame, model_id='yolox_tiny', threshold=0.25))

# Multimodal Powerhouse and LLM-based Application
chunks_t.add_embedding_index('text',
string_embed=sentence_transformer.using(model_id='intfloat/e5-large-v2'))

# Pixeltable + Next.js + FastAPI Video Frame Search Engine
@app.post("/api/search")
async def search_video(
    query: Optional[UploadFile] = File(None),
):
    try:
        frames_view = pxt.get_table('video_search.frames')

# Infinite-Memory Discord Chatbot
messages_view = self.server_tables[server_id]['messages_view']
sim = messages_view.text.similarity(query)
results_df = (
    messages_view
    .order_by(sim, asc=False)
    .select(username=messages_view.username...)
    .collect().to_pandas())
```

Pixeltable

AI DATA INFRASTRUCTURE — DECLARATIVE, MULTIMODAL, INCREMENTAL

pixeltable/pixeltable

docs.pixeltable.com

https://discord.gg/QPyqFYx2UN