2023 X:AI ADV Session

Toy Project 중간 발표

Formula To Latex .

CV2 TEAM

황건하 이승학 유광열 이서연

2023. 08. 04

TEAM MEMBER

황건하
행정학전공 16

이승학
AI빅데이터융합경영 18

유광열
AI빅데이터융합경영 19

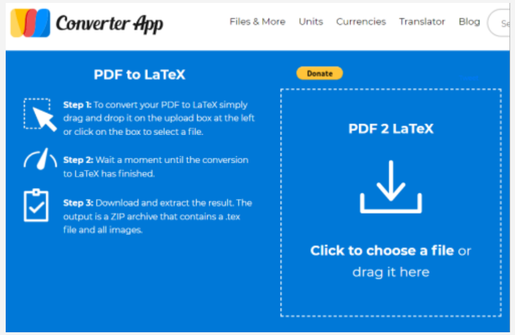
이서연
AI빅데이터융합경영 21

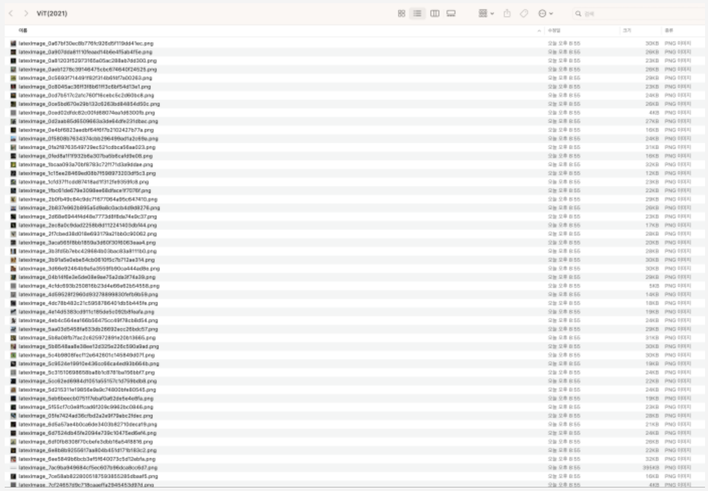
연구 배경

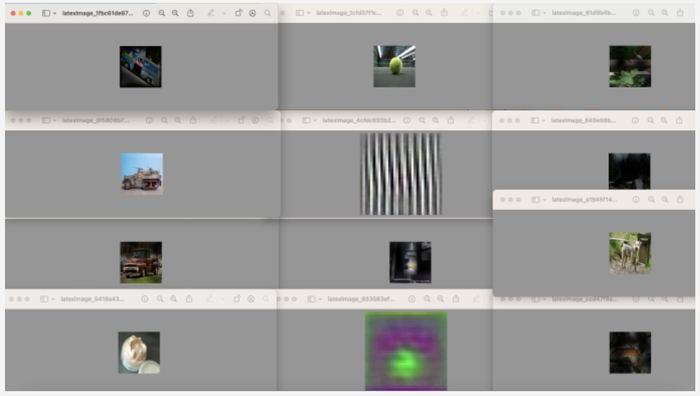
- Image To Latex의 필요성

1. 논문 번역 시 수식 변환의 어려움 존재

2. 기존에 존재하는 PDF to LaTex가 제대로 작동되지 않음







Latex 언어란?

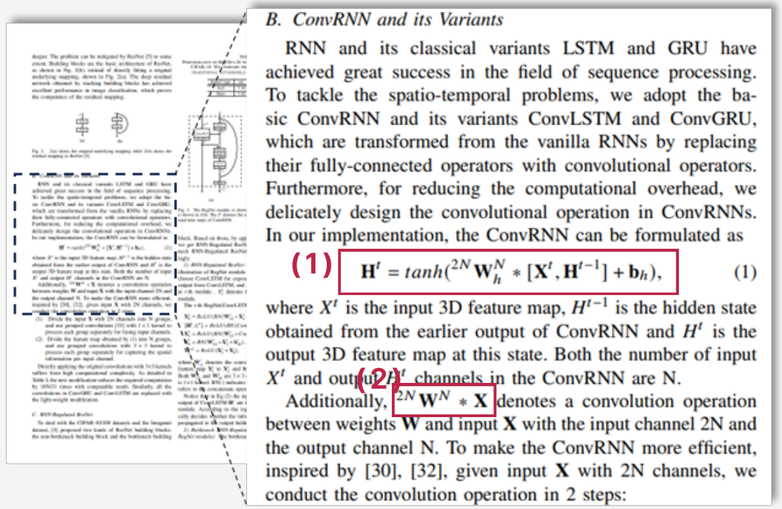
문서 작성 도구의 일종으로, 논문이나 출판물 등의 특수 형식 문서를 작성하는 데 쓰이는 시스템이며 수식, 그래프, 다이어그램을 많이 그릴 때 유용하다.

주제

이미지 속 Formula를 탐지하여 Latex로 변환되도록 하자!

Input

Formula Detection



B. ConvRNN and its Variants

RNN and its classical variants LSTM and GRU have achieved great success in the field of sequence processing. To tackle the spatio-temporal problems, we adopt the basic ConvRNN and its variants ConvLSTM and ConvGRU, which are transformed from the vanilla RNNs by replacing their fully-connected operators with convolutional operators. Furthermore, for reducing the computational overhead, we delicately design the convolutional operation in ConvRNNs. In our implementation, the ConvRNN can be formulated as

(1) $\mathbf{H}^t = tanh(^{2N}\mathbf{W}_h^N * [\mathbf{X}^t, \mathbf{H}^{t-1}] + \mathbf{b}_h),$ (1)

where X^t is the input 3D feature map, H^{t-1} is the hidden state obtained from the earlier output of ConvRNN and H^t is the output 3D feature map at this state. Both the number of input X^t and output H^t channels in the ConvRNN are N.

Additionally, (2) $^{2N}\mathbf{W}^N * \mathbf{X}$ denotes a convolution operation between weights $\mathbf{W}$ and input $\mathbf{X}$ with the input channel 2N and the output channel N. To make the ConvRNN more efficient, inspired by [30], [32], given input $\mathbf{X}$ with 2N channels, we conduct the convolution operation in 2 steps:

(1) $\mathbf{H}^t = tanh(^{2N}\mathbf{W}_h^N * [\mathbf{X}^t, \mathbf{H}^{t-1}] + \mathbf{b}_h),$

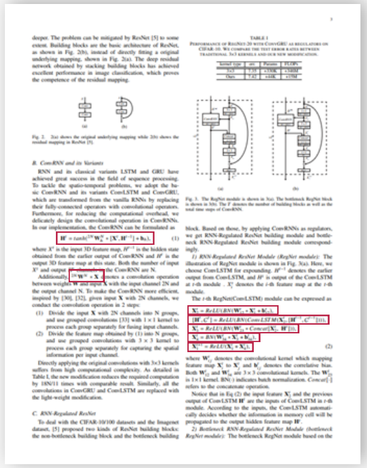
(2) $^{2N}\mathbf{W}^N * \mathbf{X}$

Image To Latex

{₩bf H}^{t}=t a n h({}^{2N}{₩bf W}_{h}^{N}₩ast[{₩bf X}^{t},{₩bf H}^{t-1}]+{₩bf b}_{h}),

^{2N}W^{N} * X

Output



No	Latex
1	{₩bf H}^{t}=t a n h({}^{2N}{₩bf W}_{h}^{N}₩ast[{₩bf X}^{t},{₩bf H}^{t-1}]+{₩bf b}_{h}),
2	^{2N}W^{N} * X

핵심 Task | Formula Detection & Image To Latex

Formula Detection

examples in the vicinity share the same class, and does not model the vicinity relation across examples of different classes.

Contribution Motivated by these issues, we introduce a simple and data-agnostic data augmentation routine, termed *mixup* (Section 2). In a nutshell, *mixup* constructs virtual training examples

$$\tilde{x} = \lambda x_i + (1-\lambda) x_j, \quad \text{where } x_i, x_j \text{ are raw input vectors}$$
$$\tilde{y} = \lambda y_i + (1-\lambda) y_j, \quad \text{where } y_i, y_j \text{ are one-hot label encodings}$$

(x_i, y_i) and (x_j, y_j) are two examples drawn at random from our training data, and $\lambda \in [0, 1]$. Therefore, *mixup* extends the training distribution by incorporating the prior knowledge that linear interpolations of feature vectors should lead to linear interpolations of the associated targets. *mixup* can be implemented in a few lines of code, and introduces minimal computation overhead.

Despite its simplicity, *mixup* allows a new state-of-the-art performance in the CIFAR-10, CIFAR-100, and ImageNet-2012 image classification datasets (Sections 3.1 and 3.2). Furthermore, *mixup* increases the robustness of neural networks when learning from corrupt labels (Section 3.4), or facing adversarial examples (Section 3.5). Finally, *mixup* improves generalization on speech (Sections 3.3) and tabular (Section 3.6) data, and can be used to stabilize the training of GANs (Section 3.7). The source-code necessary to replicate our CIFAR-10 experiments is available at:

`https://github.com/facebookresearch/mixup-cifar10`.

To understand the effects of various design choices in *mixup*, we conduct a thorough set of ablation study experiments (Section 3.8). The results suggest that *mixup* performs significantly better than related methods in previous work, and each of the design choices contributes to the final performance. We conclude by exploring the connections to prior work (Section 4), as well as offering some points for discussion (Section 5).

2 From Empirical Risk Minimization to *mixup*

In supervised learning, we are interested in finding a function $f \in \mathcal{F}$ that describes the relationship between a random feature vector X and a random target vector Y, which follow the joint distribution $P(X, Y)$. To this end, we first define a loss function ℓ that penalizes the differences between predictions $f(x)$ and actual targets y, for examples $(x, y) \sim P$. Then, we minimize the average of the loss function ℓ over the data distribution P, also known as the *expected risk*:

$$R(f) = \int \ell(f(x), y) dP(x, y).$$

Unfortunately, the distribution P is unknown in most practical situations. Instead, we usually have access to a set of training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where $(x_i, y_i) \sim P$ for all $i = 1, \dots, n$. Using the training data $\mathcal{D}$, we may approximate P by the *empirical distribution*

$$P_\delta(x, y) = \frac{1}{n} \sum_{i=1}^{n} \delta(x = x_i, y = y_i),$$

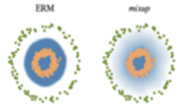
where $\delta(x = x_i, y = y_i)$ is a Dirac mass centered at (x_i, y_i). Using the empirical distribution P_δ, we can now approximate the expected risk by the *empirical risk*:

$$R_\delta(f) = \int \ell(f(x), y) dP_\delta(x, y) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i). \quad (1)$$

Learning the function f by minimizing (1) is known as the Empirical Risk Minimization (ERM) principle (Vapnik, 1998). While efficient to compute, the empirical risk (1) monitors the behaviour of f only at a finite set of n examples. When considering functions with a number parameters comparable to n (such as large neural networks), one trivial way to minimize (1) is to memorize the training data (Zhang et al., 2017). Memorization, in turn, leads to the undesirable behaviour of f outside the training data (Szegedy et al., 2014).

```
# y1, y2 should be one-hot vectors
for (x1, y1), (x2, y2) in zip(loader1, loader2):
    lam = numpy.random.beta(alpha, alpha)
    x = Variable(lam * x1 + (1. - lam) * x2)
    y = Variable(lam * y1 + (1. - lam) * y2)
    optimizer.zero_grad()
    loss(net(x), y).backward()
    optimizer.step()
```

(a) One epoch of *mixup* training in PyTorch.



(b) Effect of *mixup* ($\alpha = 1$) on a toy problem. Green: Class 0. Orange: Class 1. Blue shading indicates $p(y = 1|x)$.

Figure 1: Illustration of *mixup*, which converges to ERM as $\alpha \to 0$.

However, the naïve estimate P_δ is one out of many possible choices to approximate the true distribution P. For instance, in the *Vicinal Risk Minimization* (VRM) principle (Chapelle et al., 2000), the distribution P is approximated by

$$P_\nu(\tilde{x}, \tilde{y}) = \frac{1}{n} \sum_{i=1}^{n} \nu(\tilde{x}, \tilde{y} | x_i, y_i),$$

where ν is a *vicinity distribution* that measures the probability of finding the *virtual* feature-target pair $(\tilde{x}, \tilde{y})$ in the *vicinity* of the training feature-target pair (x_i, y_i). In particular, Chapelle et al. (2000) considered Gaussian vicinities $\nu(\tilde{x}, \tilde{y} | x_i, y_i) = \mathcal{N}(\tilde{x} - x_i, \sigma^2) \delta(\tilde{y} = y_i)$, which is equivalent to augmenting the training data with additive Gaussian noise. To learn using VRM, we sample the vicinal distribution to construct a dataset $\mathcal{D}_\nu := \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^m$, and minimize the *empirical vicinal risk*:

$$R_\nu(f) = \frac{1}{m} \sum_{i=1}^{m} \ell(f(\tilde{x}_i), \tilde{y}_i).$$

The contribution of this paper is to propose a generic vicinal distribution, called *mixup*:

$$\mu(\tilde{x}, \tilde{y} | x_i, y_i) = \frac{1}{n} \sum_{j}^{n} \mathbb{E}_\lambda [\delta(\tilde{x} = \lambda \cdot x_i + (1-\lambda) \cdot x_j, \tilde{y} = \lambda \cdot y_i + (1-\lambda) \cdot y_j)],$$

where $\lambda \sim \text{Beta}(\alpha, \alpha)$, for $\alpha \in (0, \infty)$. In a nutshell, sampling from the *mixup* vicinal distribution produces virtual feature-target vectors

$$\tilde{x} = \lambda x_i + (1-\lambda) x_j,$$
$$\tilde{y} = \lambda y_i + (1-\lambda) y_j,$$

where (x_i, y_i) and (x_j, y_j) are two feature-target vectors drawn at random from the training data, and $\lambda \in [0, 1]$. The *mixup* hyper-parameter α controls the strength of interpolation between feature-target pairs, recovering the ERM principle as $\alpha \to 0$.

The implementation of *mixup* training is straightforward, and introduces a minimal computation overhead. Figure 1a shows the few lines of code necessary to implement *mixup* training in PyTorch. Finally, we mention alternative design choices. First, in preliminary experiments we find that convex combinations of three or more examples with weights sampled from a Dirichlet distribution does not provide further gain, but increases the computation cost of *mixup*. Second, our current implementation uses a single data loader to obtain one minibatch, and then *mixup* is applied to the same minibatch after random shuffling. We found this strategy works equally well, while reducing I/O requirements. Third, interpolating only between inputs with equal label did not lead to the performance gains of *mixup* discussed in the sequel. More empirical comparison can be found in Section 3.8.

What is *mixup* doing? The *mixup* vicinal distribution can be understood as a form of data augmentation that encourages the model f to behave linearly in-between training examples. We argue that this linear behaviour reduces the amount of undesirable oscillations when predicting outside the training examples. Also, linearity is a good inductive bias from the perspective of Occam's razor,

mixup: BEYOND EMPIRICAL RISK MINIMIZATION

Image To Latex

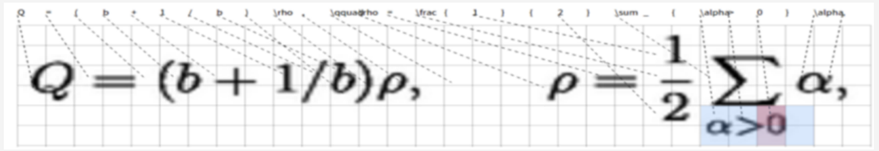
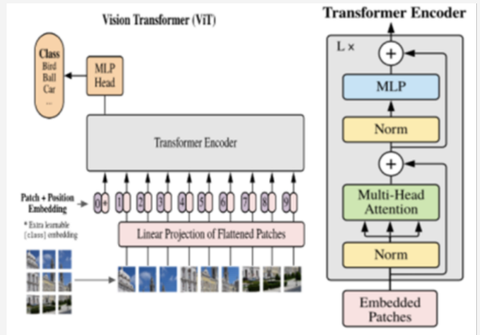
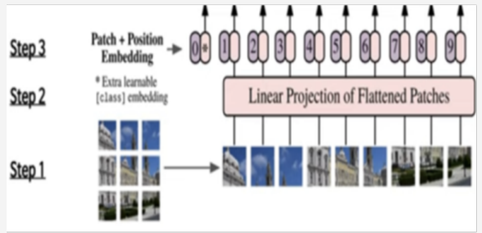
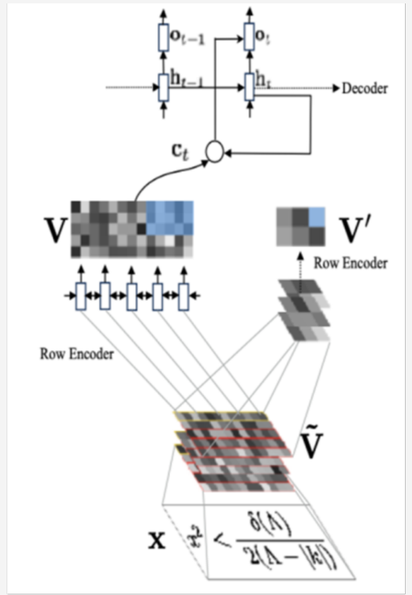


Image-To-Markup Generation with Coarse-to-Fine Attention (2017)





ViT Structure

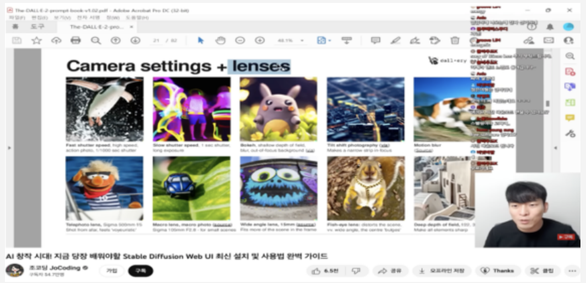


Network Structure

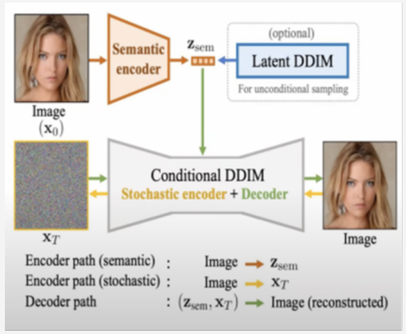
진행상황1

- Stable Diffusion

Stable Diffusion 설치 및 사용법 배우기



Diffusion Model 수학이 포함된 Tutorial 영상 참고



$$L_{t-1} = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\mu}_t(\mathbf{x}_t,\mathbf{x}_0) - \mu_\theta(\mathbf{x}_t,t)\right\|^2\right] + C$$

$$\mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t(\mathbf{x}_0,\epsilon) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) - \mu_\theta(\mathbf{x}_t(\mathbf{x}_0,\epsilon),t)\right\|^2\right]$$

$$\mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right\|^2\right]$$

Modeling Automatic1111's Stable Diffusion WebUI

진행상황1

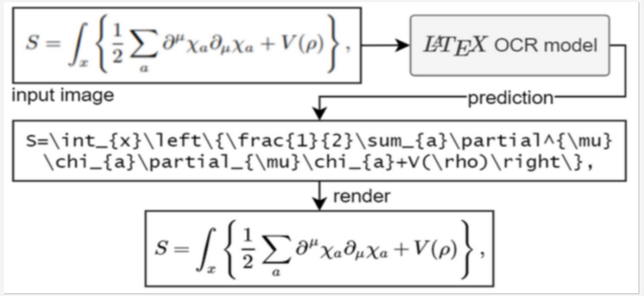
- Stable Diffusion

Modeling Automatic1111's Stable Diffusion WebUI

진행상황2

- Image To Latex

Using the Model



Data: M2LATEX-100K (Kaggle)

{ \frac { \phi ^ { \prime \prime } } { A } } + { \frac { 1 } { A } } \left(- { \frac { 1 } { 2	a4d25113b2.png	\widetilde \gamma _ { \mathrm { h o p f } } } \simeq \sum _ { n > 0 } \widetilde { G } _ { n } { \frac...	66667cee5b.png
({ \cal L } _ { a } g) _ { i j } = 0 , \ \ \ ({ \cal L } _ { a } H) _ { i j k } = 0 ,	1cbb05a562.png	\, ^ { * } d \, ^ { * } H = \kappa \, ^ { * } d \phi = J _ { B } .	242a58bc3a.png

Modeling pix2tex - LatexOCR

$$\Psi = \exp(il\varphi)\, R(\rho)\,.$$

$$\xi_n(u) = \sin\left(\frac{\pi n u}{t}\right) \qquad n = 1, 2, \ldots$$

$$\delta^V(t) g = g((t^2 - 1)\dot{X}(t) X(t)^{-1} + I),$$

$$\partial_{\vec{z}}^2 H_F + H_p \partial_{\vec{x}}^2 H_F = 0, \quad \partial_{\vec{z}}^2 H_p = 0.$$

▶▶

5abbb9b19f. png	$\int_{-c}^{\infty} dI\, e^{-Ic} \int_{-c}^{\infty} dI' e^{-I'c} II' \frac{I'-I}{I+I'} \{3\,\delta''(I) - \frac{3}{4} I\,\delta(I)\} = 0.$
329a44c373.png	$[\vec{K_a}(p), \vec{K_b}(k)] = i f_{abc} \vec{K_c}(p+k) - 2\pi^2 C p \delta_{ab} \delta_{p+k,0}.$
73b51f198b. png	$E(v) = \frac{d}{dt} E(q) \qquad \forall t\,.$
6331d9e7fd. png	$\frac{1}{L^2} \prod_{i=1}^{3} (r_+^2 + q_i) - \mu r_+^2 \sim 0\,.$
91a55d2cb9.png	$x'(\sigma + 2\pi, \tau) = x'(\sigma, \tau) + 2\pi L'.$
408fe63a30.png	$\xi_\sigma(\sigma) = 1 - \frac{\pi e^\sigma}{3} + 8\pi e^\sigma \sum_{n=1}^{\infty} \frac{n e^{-2n\pi e^\sigma}}{1 - e^{-2n\pi e^\sigma}}$
232d6fea7c. png	$m_0^2 \varphi + \frac{\mu^2 - m_0^2}{\beta} \tan(\beta\varphi) = 0$
702a6a1e50.png	$\frac{dz}{dr} = e^{2A}, \qquad \psi \to e^{-3A/2}\psi$
7a7df95199.png	$F_{A_1}(q^{a_1}, \dot{q}^{a_1}) = 0\,, \quad F_{A_0}(q^{a_1}) = 0\,.$
d39642fa83.png	$1 - \frac{\delta\phi}{2\pi} = L'(\infty) = \frac{2GW + 1}{N^2(\infty)}.$
2c65d65f05.png	$\varphi'' + \frac{1}{r}\varphi' + \lambda(\varphi - \varphi^3) = 0,$
3bea2f425a. png	$x^N + y^N = z^N, \quad \Phi(I) = \omega^{I(I+N)/2}, \quad \omega^{1/2} = \exp(\pi i/N).$
3e3f5d7c1f. png	$e^{\vec{\mu}}_{\vec{\mu}} = \begin{pmatrix} e^{\vec{z}}_{\vec{\sigma}} & 0 \\ 0 & e^{\vec{w}}_{\vec{a}} \end{pmatrix}$
12e9e7dd1f. png	$\phi = \begin{pmatrix} 0 \\ \phi_0 \end{pmatrix}$
7c99ded238.png	$\mathrm{Tr}(\mathbf{M}_+ \mathbf{M}_-) = 0\,.$
652ec3e732.png	$\int_{C_i} B_2 = 4\pi^2 \alpha' \frac{1}{2} \equiv b_0$

Progress Plan

Data Generation

기존 논문 이미지에 Formula 이미지를
Cutmix 방법을 통해 psuedo Labeling

YOLO v8

YOLO v8을 통한
Formula Detection

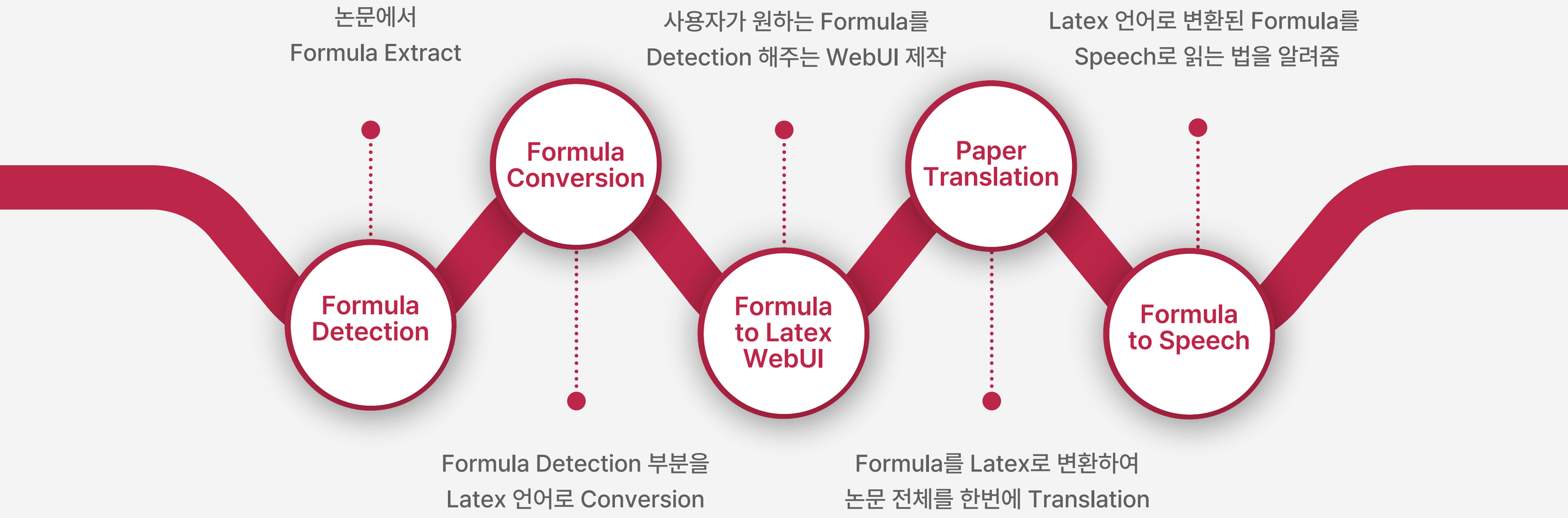
Fine Tuning

생성된 새로운 데이터로
Formula Detection

OUTPUT

Formula Detection이 된 논문 이미지
& Formula Detection Box Coordinate

Progress Plan



기대효과

논문 리뷰 작성할 때
수식 입력이 편해짐

처음보는 수학 기호에
대한 Latex를
찾아 볼 필요가 없음

전체 논문 번역 시
자연스러운 번역 결과가
나올 것으로 기대

Q & A

CV2 TEAM

황건하 이승학 유광열 이서연

2023. 08. 04