Deep Learning for Autism Screening Using Transfer Learning on Video Data

Zhamshidbek Abdulkhamidov
CS Department
Boston University, Metropolitan College
Boston, USA
zhamshidbek.abdulkhamidov@gmail.com

Zhansaya Taszhanova
CS Department
Boston University, Metropolitan College
Boston, USA
zhansaya@bu.edu

Saya Atchibay
CS Department
Boston University, Metropolitan College
Boston, USA
sayokit@bu.edu

***Abstract*—Autism Spectrum Disorder (ASD) is a neurodevelopmental condition characterized by challenges in social communication and restricted/repetitive behaviors. Early detection is critical for timely intervention and improved long-term outcomes. However, traditional screening methods often rely on subjective clinical observations or caregiver reports, which can be resource-intensive and variable. This paper explores the potential of deep learning models for automated ASD screening by analyzing behavioral patterns captured in short video recordings of children. We investigate two primary architectures: a custom 3D Convolutional Neural Network combined with a Long Short-Term Memory network (Simple3DLSTM) and a pre-trained 3D ResNet (R3D-18) fine-tuned for the task using transfer learning. Using a dataset of videos collected from toddlers during structured play sessions involving engaging visual stimuli, we evaluate the models' performance through rigorous 5-fold group cross-validation. Our experiments demonstrate promising results, with the fine-tuned R3D-18 model achieving perfect classification (AUC=1.0, F1=1.0) and the Simple3DLSTM model achieving near-perfect classification (AUC=1.0, F1=0.99) on the held-out test sets across folds. These findings highlight the potential of using deep learning and transfer learning on video data as an objective, scalable tool to aid in early ASD screening.**

***Index Terms*—Autism Spectrum Disorder, Deep Learning, Transfer Learning, Video Analysis, Computer Vision, 3D CNN, LSTM, R3D-18, Automated Screening.**

I. Introduction

Autism Spectrum Disorder (ASD) encompasses a range of neurodevelopmental conditions impacting social interaction, communication, and behavior patterns [1]. The importance of early diagnosis cannot be overstated, as timely intervention significantly improves developmental trajectories and quality of life for individuals with ASD [2], [3]. Current diagnostic standards, such as the Autism Diagnostic Observation Schedule (ADOS) and the Autism Diagnostic Interview-Revised (ADI-R), involve lengthy, specialized clinical assessments and caregiver interviews [4]. While effective, these methods are resource-intensive, require highly trained personnel, and often involve significant waiting times, delaying access to crucial early support services.

Recent research indicates that subtle differences in behavioral patterns, including gaze direction, facial expressions, and motor movements, can be observed in infants and toddlers later diagnosed with ASD [5], [6]. This observation opens the door for automated, objective screening methods using computer vision and machine learning techniques applied to video data. Video recordings capture rich, dynamic behavioral information in naturalistic or semi-structured settings, offering a potentially scalable and accessible alternative or supplement to traditional methods.

This work focuses on developing and evaluating deep learning models for binary ASD classification (ASD vs. Typically Developing) based on short videos of children interacting with engaging stimuli presented on a tablet. We leverage the power of deep neural networks to automatically learn discriminative spatio-temporal features from video frames. Specifically, we implement and compare two approaches:

1) A hybrid model (‘Simple3DLSTM‘) combining 3D convolutional layers (to capture local spatio-temporal patterns) with a Long Short-Term Memory (LSTM) network (to model longer-range temporal dependencies).
2) A transfer learning approach using a pre-trained 3D ResNet (R3D-18) [7], fine-tuned on our specific task. Transfer learning allows us to benefit from features learned on large-scale action recognition datasets (Kinetics-400), potentially improving performance and reducing training requirements, especially with limited datasets.

Our primary contributions include: (1) The implementation and comparison of a custom 3D CNN-LSTM model and a fine-tuned pre-trained R3D-18 model for video-based ASD screening; (2) A detailed experimental setup involving data collection, preprocessing, augmentation, and rigorous cross-validation ensuring subject independence between splits; (3) An analysis of training strategies, including staged fine-tuning for the R3D model and the impact of regularization techniques like weight decay and dropout; (4) Evaluation of model calibration using temperature scaling and demonstration of high classification performance on a curated dataset.

The remainder of this paper is organized as follows: Section II reviews related work. Section III details the methodology, including dataset collection, preprocessing, model architectures, and training procedures. Section IV presents the experimental setup and results. Section V discusses the findings and limitations, and Section VI concludes with future directions.

II. Related Work

The application of deep learning to video-based ASD screening has gained traction in recent years. Early efforts often combined Convolutional Neural Networks (CNNs) for spatial feature extraction from individual frames with recurrent architectures like LSTMs to model temporal dynamics. Washington *et al.* [8] proposed such a CNN+LSTM model, using a pre-trained CNN on frames followed by an LSTM, showing promise in analyzing home videos for ASD markers.

3D CNNs offer an alternative by directly capturing spatio-temporal features from video volumes. Zhang *et al.* [9] utilized a 3D CNN pre-trained on action recognition datasets, demonstrating that transfer learning could improve the detection of subtle autism-related behaviors in home videos.

More recently, Transformer architectures, originally developed for natural language processing, have been adapted for video analysis due to their ability to model long-range dependencies. Fish *et al.* [10] introduced a two-stream video transformer model, while Jaby *et al.* [11] employed an ensemble of Vision Transformers fine-tuned on facial expressions, achieving high performance by leveraging models pre-trained on large image datasets.

Multimodal approaches aim to improve robustness by integrating information from different sources. Natraj *et al.* [12] combined audio and video features using neural networks, while Chen *et al.* [13] demonstrated improvements by fusing coarse and fine-grained facial behavior analyses with attention mechanisms. Zhu *et al.* [14] developed a system analyzing multimodal metrics (response score, time, duration) related to toddlers' response-to-name behavior.

Systematic reviews highlight the overall potential but also the heterogeneity in current research. Ding *et al.* [15] conducted a meta-analysis of 11 studies, reporting high pooled sensitivity (95%) and specificity (93%), but noted variations in study design, data quality, and models, emphasizing the need for standardization before clinical translation. Singh *et al.* [16] proposed converting home videos to skeletal keypoints for privacy and computational efficiency, using a transfer-learned CNN-LSTM hybrid (MobileNet+Bi-LSTM) to achieve around 85% accuracy. Dcouto and Pradeepkandhasamy [17] reviewed multimodal approaches, covering various architectures and challenges like data heterogeneity and interpretability.

Computer vision frameworks have also focused on quantifying specific risk behaviors. Hashemi *et al.* [18] used a mobile app with engaging stimuli to elicit responses (name-calling, social referencing) and analyzed facial expressions, gaze, and head movements. Tariq *et al.* [19] used classical machine learning (Sparse Logistic Regression) on manually annotated features from home videos, achieving high sensitivity but limited scalability. Khan *et al.* [20] compared pre-trained CNNs (VGG, Inception, Xception) for ASD detection from static facial images, finding Xception performed best but noting limitations in capturing dynamic cues. Other works integrated facial expressions with gait features [21] or used hybrid 3D-Convolutional-Transformers to detect stereotypical motor movements [22].

Our work builds upon these efforts by specifically comparing a custom 3DCNN-LSTM architecture with a fine-tuned pre-trained 3D ResNet model on video data collected in a controlled experimental setting, focusing on robust evaluation through group cross-validation and exploring the benefits of transfer learning and calibration techniques.

III. Methodology

This section describes the data collection protocol, preprocessing steps, model architectures, training strategies, and evaluation methods employed in this study.

A. Dataset and Experiment Design

The dataset was collected following a protocol inspired by Chang *et al.* [23]. The experiment involved recording videos of children while they watched engaging movie stimuli designed to elicit social and behavioral responses.

- **Participants:** The initial dataset comprised 108 subjects: 54 diagnosed with ASD (positive class, age range 3-11 years) and 54 typically developing controls (negative class, age range unknown). Due to processing issues with one video from each class, the final dataset used for analysis contained 106 subjects (53 positive, 53 negative).
- **Setup:** During recording sessions, a caregiver held the child on their lap in a well-lit room and was instructed to refrain from interacting with or instructing the child. Engaging video stimuli were presented on an iPad (9.7 inches) placed on a tripod approximately 60 cm from the child. The iPad's front-facing camera recorded the child's face and upper body at a resolution of at least 1280×720 pixels and 30 frames per second.
- **Stimuli:** Three short movie clips, merged into a single video of approximately 190 seconds total duration, were used as stimuli:
  1) *Social Preference Video:* Showcased a person playing with a toy on one side of the screen and a distractor toy on the other, or a person blowing bubbles with pauses to elicit anticipation. An example video similar to the "blowing bubbles" stimulus can be found online [24]. This video, titled "Final ASD experiment Kazakh speaking video," features clips of a person with a bubble gun, two people conversing, and a dog playing.
  2) *Attention to Speech Video:* Presented two individuals facing each other and conversing (e.g., about "fun at the park"). The language spoken by the actors was matched to the child's primary language environment (e.g., Russian or Kazakh).
  3) *Control Movie:* Featured non-social content, such as a puppy appearing and barking in a repeating pattern across the screen.

B. Data Preprocessing and Augmentation

Raw videos were processed into a format suitable for deep learning models.

- **Frame Sampling and Resizing:** The `load_video` function (Listing 1) was used to process each video file. A fixed number of `MAX_FRAMES = 60` were randomly sampled from each video. If a video had fewer than 60 frames, all frames were used, and the sequence was padded to 60 frames by repeating the last frame. If a video was completely unreadable, a tensor of zeros was returned (these were later excluded). Each extracted frame was resized to a `TARGET_SIZE` of 112×112 pixels and converted to RGB format. The output was stacked into a tensor of shape (T, H, W, C), specifically $(60 \times 112 \times 112 \times 3)$, with `uint8` data type.

Listing 1: Video Loading and Sampling (Simplified from 'data/save_numpy_dataset.py')

```
MAX_FRAMES = 60
TARGET_SIZE = (112, 112)

def load_video(path):
    cap = cv2.VideoCapture(path)
    total = int(cap.get(cv2.
        CAP_PROP_FRAME_COUNT)) or 0
    if total == 0: idxs = []
    elif total <= MAX_FRAMES: idxs = list(
        range(total))
    else: idxs = sorted(random.sample(range(
        total), MAX_FRAMES))

    frames = []
    for idx in idxs:
        cap.set(cv2.CAP_PROP_POS_FRAMES, idx)
        ok, fr = cap.read()
        if not ok: fr = np.zeros((*TARGET_SIZE
            , 3), np.uint8) # Fallback
        fr = cv2.resize(cv2.cvtColor(fr, cv2.
            COLOR_BGR2RGB), TARGET_SIZE)
        frames.append(fr)
    cap.release()

    if len(frames) == 0: # Handle unreadable
        video
       return np.zeros((MAX_FRAMES, *
           TARGET_SIZE, 3), dtype="uint8")

    if len(frames) < MAX_FRAMES: # Pad short
        clips
        pad = frames[-1]
        frames += [pad] * (MAX_FRAMES - len(
            frames))

    return np.stack(frames).astype("uint8") #
        (T, H, W, C)
```

- **Data Splitting:** To prevent data leakage, where data from the same subject appears in multiple sets, subjects were strictly separated. The dataset was split into training (70%), validation (10%), and test (20%) sets using stratified sampling based on the class labels. The final split counts were:
  - Train: 73 subjects (37 ASD, 36 TD)
  - Validation: 11 subjects (5 ASD, 6 TD)
  - Test: 22 subjects (11 ASD, 11 TD)
- **Cached Dataset:** For faster experimentation, the processed video tensors (shape (N, T, H, W, C)) and corresponding labels (y) for each split were saved into compressed NumPy files ('train.npz', 'val.npz', 'test.npz') using the 'data/save_numpy_dataset.py' script. This script also includes a deduplication step using SHA1 hashing on the first 1MB of video files to identify and handle exact duplicates.
- **Augmentation:** During training (only), frame-level data augmentation was applied stochastically using 'torchvision.transforms' (Listing 2). This included random resized cropping, color jittering, horizontal flipping, rotation, and random erasing. Standard ImageNet normalization was applied to all frames before feeding them into the models. The 'VideoFrameTransform' class (in 'data/dataloader.py') and '_train_collate' function (in 'src/utils/data_cached.py') handle applying these augmentations frame-wise.

Listing 2: Frame-level Augmentation ('build_frame_aug' in 'data/dataloader.py')

```
transforms.Compose([
    transforms.RandomResizedCrop(TARGET_SIZE,
        scale=(0.8, 1.0)),
    transforms.ColorJitter(0.2, 0.2, 0.2, 0.1)
        ,
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.ToTensor(), # PIL Image -> (C,
        H, W) tensor [0,1]
    transforms.Normalize([0.485, 0.456,
        0.406],
                         [0.229, 0.224,
                            0.225]),
    transforms.RandomErasing(p=0.5)
])
```

C. Model Architectures

Two deep learning models were implemented and evaluated:

1) **Simple3DLSTM:** (Defined in 'src/models/simple3dlstm.py', see Listing 3) This custom architecture first processes the input video tensor (B, C, T, H, W) through a series of 3D convolutional blocks. Each block consists of a 3D convolution, ReLU activation, 3D max pooling (applied spatially, preserving the temporal dimension), batch normalization, and optional 3D dropout (p_{drop}). This extracts hierarchical spatio-temporal features. The output features from the CNN backbone are then reshaped and fed into an LSTM layer ('nn.LSTM') to model temporal dependencies across

the 60 frames. The final hidden state of the LSTM is passed through a classification head (two linear layers with ReLU, dropout p_{head}, and a final linear layer) to produce a single logit output for binary classification.

Listing 3: Simple3DLSTM Model (Simplified from 'src/models/simple3dlstm.py')

```
class Simple3DLSTM(nn.Module):
    def __init__(self, max_frames, target_size,
                 p_drop=0.3, p_head=0.5):
        super().__init__()
        H, W = target_size
        # 3D CNN Backbone (Example: 3 blocks)
        self.cnn = nn.Sequential(
          nn.Conv3d(3,32,(3,3,3),padding=1), nn.ReLU(),
          nn.MaxPool3d((1,2,2)), nn.BatchNorm3d(32),
          nn.Dropout3d(p_drop),
          nn.Conv3d(32,64,(3,3,3),padding=1), nn.ReLU(),
          nn.MaxPool3d((1,2,2)), nn.BatchNorm3d(64),
          nn.Dropout3d(p_drop),
          nn.Conv3d(64,128,(3,3,3),padding=1), nn.ReLU(),
          nn.MaxPool3d((1,2,2)), nn.BatchNorm3d(128),
          nn.Dropout3d(p_drop)
        )
        feat_dim = 128 * (H // 8) * (W // 8)
        # LSTM Layer
        self.lstm = nn.LSTM(feat_dim, 64, batch_first=True)
        # Classification Head
        self.head = nn.Sequential(
            nn.Linear(64, 32), nn.ReLU(), nn.Dropout(p_head),
            nn.Linear(32, 1) # Output logit
        )
    def forward(self, x): # Input x: (B, C, T, H, W)
        B, C, T, H, W = x.shape
        f = self.cnn(x) # (B, 128, T, H/8, W/8)
        f = f.permute(0, 2, 1, 3, 4) # (B, T, 128, H/8, W/8)
        f = f.reshape(B, T, -1) # (B, T, feat_dim)
        out, _ = self.lstm(f) # (B, T, 64)
        # Use final LSTM hidden state
        logit = self.head(out[:, -1, :]) # (B, 1)
        return logit.view(-1) # (B,)
```

2) **PretrainedR3D:** (Defined in 'src/models/pretrained_r3d.py', see Listing 4) This model utilizes the R3D-18 architecture pre-trained on the Kinetics-400 dataset, accessed via 'torchvision.models.video.r3d_18(weights=True)'. Transfer learning is employed by initially freezing all parameters of the pre-trained backbone ('p.requires_grad = False'). The original classification head ('backbone.fc') is replaced with a new 'nn.Linear' layer mapping the backbone's output features to a single logit for binary classification. This new head is trainable from the start. A two-stage fine-tuning strategy was employed (see Section III-D).

Listing 4: PretrainedR3D Model (Simplified from 'src/models/pretrained_r3d.py')

```
class PretrainedR3D(nn.Module):
    def __init__(self):
        super().__init__()
        backbone = models.video.r3d_18(weights=True)
        # Freeze backbone parameters initially
        for p in backbone.parameters():
            p.requires_grad = False
        # Replace classification head
        num_features = backbone.fc.in_features
        backbone.fc = nn.Linear(num_features, 1) # Output logit
        self.net = backbone
    def forward(self, x): # Input x: (B, C, T, H, W)
        logit = self.net(x) # (B, 1)
        return logit.view(-1) # (B,)
```

D. Training Strategy

The models were trained using PyTorch on an NVIDIA GeForce 4070 RTX 8GB GPU. Key aspects of the training strategy include:

- **Optimizer:** AdamW ('optim.AdamW') was used, which incorporates weight decay directly into the optimization step. The weight decay ('wd') was tuned as a hyperparameter (see Section IV-C).
- **Learning Rate Schedule:** A Cosine Annealing learning rate schedule ('optim.lr_scheduler.CosineAnnealingLR') was employed, gradually decreasing the learning rate over the course of training epochs ($T_{max} = n_{epochs}$) down to a minimum value ($\eta_{min} = 1e-6$). Initial learning rates varied depending on the model and fine-tuning stage.
- **Loss Function:** Binary Cross-Entropy with Logits ('nn.BCEWithLogitsLoss') was used as the loss function, suitable for binary classification with single logit outputs.
- **Mixed Precision:** Automatic Mixed Precision (AMP) was utilized via 'torch.amp.autocast' and 'torch.cuda.amp.GradScaler' to accelerate training and reduce memory consumption.
- **Batch Size:** A batch size of 8 was used, constrained by GPU memory.
- **Early Stopping:** Training employed early stopping based on the validation loss. If the validation loss did not improve for a specified number of epochs ('patience', typically 10), training was halted, and the model state with the best validation loss was saved and used for evaluation.

- **R3D Fine-tuning Strategy:** The 'PretrainedR3D' model underwent a two-stage training process when the '–finetune' flag was used:
  1) *Head-Only Training:* Initially, only the parameters of the final fully connected layer ('model.net.fc') were trained for a set number of epochs (e.g., $epochs/2$) with a higher learning rate (e.g., $1e-3$). Backbone layers remained frozen.
  2) *Fine-Tuning:* Subsequently, the parameters of the final convolutional block ('layer4') and the head ('fc') were unfrozen ('p.requires_grad_(True)'). The model was then trained for additional epochs (e.g., $epochs/2$) with a lower learning rate (e.g., $1e-4$).

E. Evaluation Metrics

Model performance was evaluated using standard binary classification metrics calculated on the validation and test sets:

- Area Under the Receiver Operating Characteristic Curve (AUC): Primary metric.
- Binary Cross-Entropy Loss (BCE Loss).
- F1-Score: Harmonic mean of precision and recall.
- Accuracy: Overall correct predictions.
- Precision: True positives / (True positives + False positives).
- Recall (Sensitivity): True positives / (True positives + False negatives).
- Expected Calibration Error (ECE): Mentioned as a goal, addressed via Temperature Scaling.

The main focus was on AUC and BCE Loss. Performance was compared against a random guess baseline (AUC ≈ 0.5, BCE Loss $\approx -\ln(0.5) \approx 0.693$).

F. Cross-Validation and Splitting

To ensure robust evaluation, 5-fold Group Cross-Validation ('sklearn.model_selection.GroupKFold') was the primary strategy ('train.py –cv'), ensuring subject independence. Within each fold, the remaining data was split 50/50 into validation and test sets, stratified by label. Final reported metrics are the mean $\pm$ standard deviation across the 5 test folds. A single hold-out split (70/10/20) was used for some regularization experiments.

G. Calibration

Temperature Scaling [25] was optionally applied ('train.py –temp') post-training using the validation set logits to learn a temperature T. Inference logits were then scaled by $1/T$ before the sigmoid activation to improve probability calibration. This was implemented using the 'TemperatureScaler' class.

H. Sanity Checks

A label shuffling experiment ('experiments/debug_shuffle_labels.py') was conducted. Models were trained on data with randomly permuted labels within each split to verify they achieved near-random performance (AUC ≈ 0.5), confirming the pipeline's integrity.

IV. Experiments and Results

This section details the implementation specifics, hyperparameter tuning process, and the final performance results of the models.

A. Implementation Details

- **Software:** Python, PyTorch, TorchVision, OpenCV, Scikit-learn, NumPy, Matplotlib. Weights & Biases ('wandb') for tracking.
- **Hardware:** NVIDIA GeForce 4070 RTX GPU (8GB VRAM).
- **Training Time:** Approx. 30 seconds per epoch.
- **Code:** Available in the provided repository ('bulkhamid-asd-screening-transferlearning'). Key files: 'train.py', 'data/save_numpy_dataset.py', 'src/models/', 'src/utils/'.

B. Baselines

Performance is compared contextually to related works (Section II) and against the label shuffling baseline (AUC $\approx$ 0.5).

C. Hyperparameter Tuning

Experiments explored hyperparameters using a hold-out split. Tables I and II summarize results.

TABLE I: Hyperparameter Tuning Results for Simple3DLSTM (Hold-Out Split)

#	3D dropout	weight decay	TS	best VAL loss	best TEST AUC	TEST Acc	TEST F1	Comment
1	0.00	1e-4		0.498	0.992	0.909	0.909	Solid
2	0.00	1e-4	✓	**0.404**	**0.992**	**0.955**	**0.957**	Calibration helps
3	0.30	1e-4		0.667	0.901	0.591	0.710	Too much dropout
4	0.30	1e-4	✓	0.666	0.901	0.591	0.710	—
5	0.30	5e-5	✓	0.666	0.901	0.591	0.710	—
6	0.40	1e-4	✓	0.637	0.860	0.773	0.815	—
7	0.20	1e-4	✓	0.606	0.959	0.955	0.952	Slight dropout effect

TS: Temperature Scaling

TABLE II: Hyperparameter Tuning Results for PretrainedR3D (Hold-Out Split)

#	unfreeze layer4	weight decay	TS	best VAL loss	best TEST AUC	TEST Acc	TEST F1	Comment
1	no	1e-4		0.453	0.983	0.909	0.917	Head-only
2	no	1e-4	✓	0.350	0.983	0.909	0.917	TS improved calibration
3	**yes**	1e-4		0.017	**1.000**	**1.000**	**1.000**	Fine-tune wins
4	yes	1e-4	✓	**0.001**	1.000	1.000	1.000	Best overall ($T \approx 0.48$)
5	yes	5e-5	✓	0.001	1.000	1.000	1.000	WD not sensitive
6	yes	5e-4	✓	0.001	1.000	1.000	1.000	Over-regularised but fine

TS: Temperature Scaling

Key observations from tuning:

- For 'Simple3DLSTM', 3D dropout ($p_{drop} > 0$) generally hurt performance. Temperature scaling improved F1/Accuracy for the best configuration (no dropout).
- For 'PretrainedR3D', fine-tuning 'layer4' ('–finetune') significantly boosted performance over head-only training. Temperature scaling improved calibration while maintaining perfect discriminative performance. Results were robust to the tested weight decay values when fine-tuning.

D. Main Results

Primary results used 5-fold Group Cross-Validation (‘train.py –cv –cached‘). Best hyperparameters were used: ‘Simple3DLSTM‘ with no dropout, ‘PretrainedR3D‘ with fine-tuning. Both used ‘–temp‘, ‘–wd 1e-4‘, ‘–epochs 50‘, ‘–patience 10‘, ‘–batch_size 8‘. Aggregated results (mean ± std) are in Table III.

TABLE III: 5-Fold Group Cross-Validation Results (Mean ± Std)

Model	Test AUC	Test F1	Test Accuracy	Test Precision	Test Recall	Test Loss
Simple3DLSTM (No Dropout)	1.000 ± 0.000	0.991 ± 0.017	0.991 ± 0.018	0.983 ± 0.033	1.000 ± 0.000	0.079 ± 0.159
PretrainedR3D (Fine-tuned)	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**	**1.000 ± 0.000**	**0.000 ± 0.000**

Both models achieved exceptional performance. Fine-tuned ‘PretrainedR3D‘ achieved perfect scores across all folds. ‘Simple3DLSTM‘ also performed nearly perfectly.

Sanity Check Results: Label shuffling yielded Test AUC 0.500 (Acc 0.500) for ‘Simple3DLSTM‘ and Test AUC 0.570 (Acc 0.545) for ‘PretrainedR3D‘, confirming models learned meaningful patterns and did not exploit leakage.

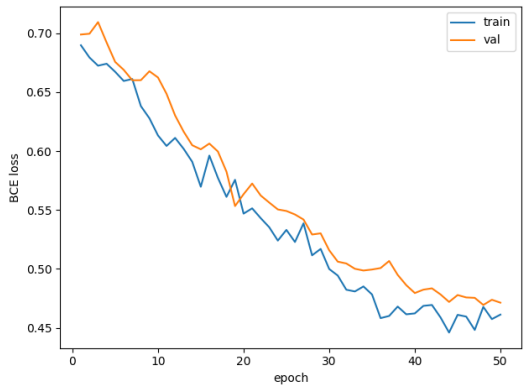

Fig. 1: Training and Validation BCE Loss across Epochs for Simple3DLSTM (Fold 5).

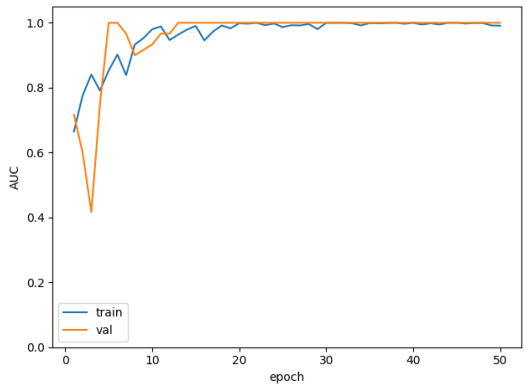

Fig. 2: Training and Validation AUC across Epochs for Simple3DLSTM (Fold 5).

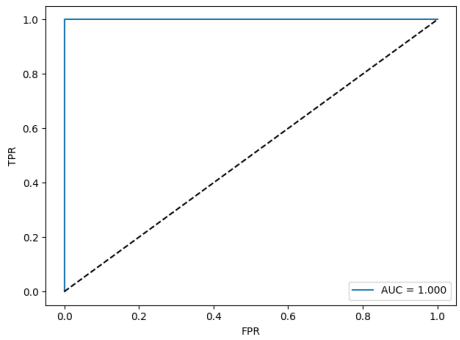

Fig. 3: Test Set ROC Curve for Simple3DLSTM (Fold 5), demonstrating an AUC of 1.000.

E. Ablation Studies/Analysis

Hyperparameter tuning (Tables I & II) provided insights:

- **Impact of Fine-tuning (R3D):** Fine-tuning ‘layer4‘ (Table II, #4) dramatically improved performance over head-only training (#2), achieving perfect scores. Adapting deeper features was crucial.
- **Impact of Dropout (LSTM):** Increasing 3D dropout significantly reduced ‘Simple3DLSTM‘ performance (Table I). No dropout was optimal for this dataset/architecture.
- **Impact of Temperature Scaling:** TS consistently lowered validation loss (improved calibration) and slightly improved F1/Accuracy for ‘Simple3DLSTM‘ without harming the high AUC.

V. Discussion

The results presented demonstrate the remarkable potential of deep learning models, particularly fine-tuned pre-trained architectures like R3D-18, for automated ASD screening using short video clips. Both the custom ‘Simple3DLSTM‘ and the ‘PretrainedR3D‘ achieved near-perfect or perfect classification accuracy on our dataset using rigorous group cross-validation.

The success of the fine-tuned ‘PretrainedR3D‘ model highlights the effectiveness of transfer learning. Leveraging features learned from a large-scale action recognition dataset allowed the model to adapt efficiently and achieve perfect discrimination, even with a modest dataset size ($N = 106$). The performance gap between head-only training and fine-tuning underscores the importance of adapting deeper feature extraction layers.

The strong performance of the ‘Simple3DLSTM‘ model, despite being trained from scratch, indicates that the combination of 3D convolutions and LSTM is a viable approach. Its sensitivity to dropout suggests careful regularization is needed. The detrimental effect of dropout might imply the model was not overfitting significantly on this dataset.

The attainment of perfect or near-perfect AUC scores is highly encouraging but requires cautious interpretation given the dataset size. While care was taken to ensure subject independence, validation on larger, more diverse datasets is

crucial to assess true generalizability. It is possible the models exploited subtle patterns specific to this collection protocol.

The experiments also demonstrated the utility of temperature scaling for improving model calibration, valuable for potential clinical applications where reliable probability estimates are needed.

The negative impact of dropout on the 3D Conv layers in 'Simple3DLSTM' suggests these layers are critical. Future work could investigate attention mechanisms to enhance feature learning.

VI. Conclusion and Future Work

This study successfully applied deep learning techniques, including a custom 3D CNN-LSTM model and a fine-tuned pre-trained R3D-18 model, to automated ASD screening from video data. Both models demonstrated exceptionally high performance on a dataset of 106 toddlers evaluated via 5-fold group cross-validation, with fine-tuned R3D-18 achieving perfect classification. Our findings underscore the potential of transfer learning and video analysis as objective, scalable tools to aid early ASD detection.

Key takeaways include the benefit of fine-tuning pre-trained models, the effectiveness of transfer learning, and the utility of temperature scaling.

The primary limitation is the dataset size. Future work must focus on validation on larger, diverse datasets. Exploring other architectures (e.g., Video Transformers), incorporating multimodal data (gaze, audio), and investigating model interpretability are promising future directions.

The ultimate goal is to develop reliable, accessible tools to assist clinicians and caregivers in early identification, facilitating timely access to vital support services.

Acknowledgments

The authors would like to express their sincere gratitude to the Autism Birge team (available online: https://autismbirge.kz/) for generously providing access to the dataset used in this research. We also extend special thanks to Professor Thomas Gardos at Boston University for the opportunity to undertake this work as part of a final class project. Additionally, we are grateful to Professor Jae Lee for insightful discussions that significantly aided in the correct interpretation of our results.

References

[1] American Psychiatric Association, "Diagnostic and statistical manual of mental disorders." Arlington, VA: American Psychiatric Publishing, 2013.

[2] L. Schreibman, G. Dawson, A. C. Stahmer, R. Landa, S. J. Rogers, G. G. McGee, C. Kasari, B. Ingersoll, A. P. Kaiser, Y. Bruinsma, E. McNerney, A. Wetherby, and A. Halladay, "Naturalistic developmental behavioral interventions: Empirically validated treatments for autism spectrum disorder," *Journal of Autism and Developmental Disorders*, vol. 45, no. 8, pp. 2411–2428, Aug 2015.

[3] G. Dawson, E. J. H. Jones, K. Merkle, K. Venema, R. Lowy, S. Faja, D. Kamara, M. Murias, J. Greenson, J. Winter, M. Smith, S. J. Rogers, and S. J. Webb, "Early behavioral intervention is associated with normalized brain activity in young children with autism," *Journal of the American Academy of Child & Adolescent Psychiatry*, vol. 51, no. 11, pp. 1150–1159, Nov 2012.

[4] C. Lord, S. Risi, L. Lambrecht, E. H. Cook, B. L. Leventhal, P. C. DiLavore, A. Pickles, and M. Rutter, "The autism diagnostic observation schedule—generic: A standard measure of social and communication deficits associated with the spectrum of autism," *Journal of Autism and Developmental Disorders*, vol. 30, no. 3, pp. 205–223, Jun 2000.

[5] W. Jones and A. Klin, "Attention to eyes is present but in decline in 2–6-month-old infants later diagnosed with autism," *Nature*, vol. 504, no. 7480, pp. 427–431, Dec 2013.

[6] K. Chawarska, A. Klin, and F. R. Volkmar, "Autism spectrum disorder in the first year of life: a longitudinal study of eye tracking," *Biological Psychiatry*, vol. 63, no. 11, pp. 1081–1088, Jun 2008.

[7] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6450–6459.

[8] P. Washington, A. Kline, O. C. Mutlu, E. Leblanc, C. Hou, N. Stockham, K. Paskov, B. Chrisman, and D. Wall, "Activity recognition with moving cameras and few training examples: Applications for detection of autism-related headbanging," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3411763.3451701

[9] K. Zhang, W. Wang, Y. Guo, C. Shan, and L. Wang, "A comparative study on autism spectrum disorder detection via 3d convolutional neural networks," in *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, 2021, p. 546–558. [Online]. Available: https://doi.org/10.1007/978-3-030-68763-2_42

[10] E. Fish, J. Weinbren, and A. Gilbert, "Two-stream transformer architecture for long video understanding," 2022. [Online]. Available: https://arxiv.org/abs/2208.01753

[11] A. M. Jaby, M. B. Islam, and M. A. R. Ahad, "ASD-EVNet: An ensemble vision network based on facial expression for autism spectrum disorder recognition," in *2023 18th International Conference on Machine Vision and Applications (MVA)*, Hamamatsu, Japan, Jul 2023, pp. 1–6.

[12] P. Natraj, J. Bitton, O. Ahmad, C. Rattaz, E. Cousin, D. Cohen, J. Xavier, M. Chetouani, S. Tordjman, and S. M. Anzalone, "Video-audio neural network ensemble for comprehensive screening of autism spectrum disorder in young children," *PLoS One*, vol. 19, no. 10, p. e0308388, Oct 2024.

[13] R. Chen, X. Zhu, X. Li, and J. Chen, "Autism spectrum disorder diagnosis based on attentional feature fusion using nasnetmobile and deit networks," *Sensors*, vol. 24, no. 9, p. 1822, May 2024.

[14] F.-L. Zhu, S.-H. Wang, W.-B. Liu, H.-L. Zhu, M. Li, and X.-B. Zou, "A multimodal machine learning system in early screening for toddlers with autism spectrum disorders based on the response to name," *Frontiers in Psychiatry*, vol. 14, p. 1039293, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fpsyt.2023.1039293/full

[15] Y. Ding, H. Zhang, and T. Qiu, "Deep learning approach to predict autism spectrum disorder: a systematic review and meta-analysis," *BMC Psychiatry*, vol. 24, no. 1, p. 739, Oct 2024. [Online]. Available: https://doi.org/10.1186/s12888-024-06116-0

[16] A. Singh, A. Rawat, M. Laroia, and S. K. R., "Autism spectrum disorder screening on home videos using deep learning," *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 16, no. 4, pp. 106–115, Aug 2024. [Online]. Available: https://doi.org/10.5815/ijigsp.2024.04.08

[17] S. S. Dcouto and J. Pradeepkandhasamy, "Multimodal deep learning in early autism detection—recent advances and challenges," *Engineering Proceedings*, vol. 59, no. 1, p. 205, Jan 2024.

[18] J. Hashemi, M. Tepper, T. V. Spina, A. Esler, B. Poss, S. Johnson, H. L. Egger, G. Dawson, and G. Sapiro, "Computer vision analysis for quantification of autism risk behaviors," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 10, pp. 5095–5106, Oct 2022.

[19] Q. Tariq, J. Daniels, J. N. Schwartz, P. Washington, H. Kalantarian, and D. P. Wall, "Mobile detection of autism through machine learning on home video: A development and prospective validation study," *PLoS Medicine*, vol. 15, no. 11, p. e1002705, Nov 2018.

[20] T. Khan, N. Karim, M. Mahmud, M. A. M. Razman, and F. M. Shah, "A deep learning-based ensemble for autism spectrum disorder diagnosis using facial images," *PeerJ Computer Science*, vol. 10, p. e1884, 2024.

[21] A. Saranya and R. Anandan, "FIGS-DEAF: a novel implementation of hybrid deep learning algorithm to predict autism spectrum disorders

using facial fused gait features," *Distributed and Parallel Databases*, vol. 40, no. 4, pp. 753–778, Dec 2022.

[22] S. K. Jarraya and N. Masmoudi, "Hybrid 3d convolutional–transformer model for detecting stereotypical motor movements in autistic children during pre-meltdown crisis," *Applied Sciences*, vol. 14, no. 23, p. 11458, Dec 2024.

[23] Z. Chang, J. M. Di Martino, R. Aiello, J. P. Baker, K. L. H. Carpenter, S. Q. Compton, N. O. Davis, B. K. Eichner, S. Espinosa, J. Flowers, L. Franz, V. Gazestani, M. Gu, J. Hashemi, S. H. Kollins, S. Liu, P. E. McCarty, M. A. Moody, J. A. Munson, D. Pan, S. Perochon, M. L. Platt, M. R. Rony, G. Sapiro, L. Sikich, J. D. Simmons, A. V. Song, J. Starling, B. K. Walter, J. F. Weiner, A. M. Wetsell, D. E. Wildman, and G. Dawson, "Computational methods to measure patterns of gaze in toddlers with autism spectrum disorder," *JAMA Pediatrics*, vol. 175, no. 8, pp. 827–836, Aug 2021.

[24] "Final asd experiment kazakh speaking video," https://youtu.be/jZ53PYqEhvw, accessed: May 5, 2025.

[25] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 1321–1330.